WAVELET-BASED IMAGE COMPRESSION FOR MOBILE APPLICATIONS

A thesis submitted to Middlesex University
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Vooi Voon YAP

School of Computing Science

Middlesex University

2005

Abstract

The transmission of digital colour images is rapidly becoming popular on mobile telephones, Personal Digital Assistant (PDA) technology and other wireless based image services. However, transmitting digital colour images via mobile devices is badly affected by low air bandwidth. Advances in communications channels (example 3G communication network) go some way to addressing this problem but the rapid increase in traffic and demand for ever better quality images, means that effective data compression techniques are essential for transmitting and storing digital images. The main objective of this thesis is to offer a novel image compression technique that can help to overcome the bandwidth problem. This thesis has investigated and implemented three different wavelet-based compression schemes with a focus on a suitable compression method for mobile applications.

The first described algorithm is a dual wavelet compression algorithm, which is a modified conventional wavelet compression method. The algorithm uses different wavelet filters to decompose the luminance and chrominance components separately. In addition, different levels of decomposition can also be applied to each component separately. The second algorithm is segmented wavelet-based, which segments an image into its smooth and non-smooth parts. Different wavelet filters are then applied to the segmented parts of the image. Finally, the third algorithm is the hybrid wavelet-based compression system (HWCS), where the subject of interest is cropped and is then compressed using a wavelet-based method. The details of the background are reduced by averaging it and sending the background separately from the compressed subject of interest. The final image is reconstructed by replacing the averaged background image pixels with the compressed cropped image.

For each algorithm the experimental results presented in this thesis clearly demonstrated that encoder output can be effectively reduced while maintaining an acceptable image visual quality particularly when compared to a conventional wavelet-based compression scheme.

Acknowledgements

I like to take this opportunity to express my deepest gratitude to my supervisor Professor Richard Comley for his enthusiasm, advice, encouragement and support, which he has shown for the work which I have performed under his supervision. Thanks are also due to Dr. Xiahong Gao and Dr. Shahedur Rahman for discussions we had while I was working on my thesis.

This work would not be possible without the generous PhD studentship from Middlesex University, which I gratefully acknowledge.

I would like to thank several of my PhD colleagues for their encouragement and faith in me while was working on my thesis.

I would like also to thank my wife, Sumathi, for her love, encouragement and support she has shown, which I could not have done without. Finally, a special thanks to my son, Chi-liq, who is a constant inspiration to me.

Table of Contents

Chapter 6

Approaches to Compressing Grey-Scale and Colour Images

Chapter 7

A Hybrid Wavelet-based Compression System

List of Figures

List of Tables

Abbreviations

ID DWT	one-dimensional wavelet transform
3G	third generation
ADSL	Asymmetric Digital Subscriber Line
AI	artificial intelligent
AM	average magnitude
BMW	balanced multiwavelets
bpp	bits-per-pixel
CEZW	Colour Embedded Zerotree Wavelet
CIE	Commission International de l'Eclairage - the International Commission on Illumination
CR	compression ratio
CSF	contrast sensitivity function
CSPIHT	Colour SPIHT
CWT	continuous wavelet transform
CYM	cyan, yellow, magenta
CZW	CEZW SOT
DCT	discrete cosine transform
DFT	discrete Fourier transform
DOF	depth of field
DPCM	differential PCM
DSL	Digital Subscriber Line
DVD	Digital Versatile Disk
DWT	discrete wavelet transforms
EBCOT	Embedded Block Coding with Optimal Truncation
EI	International hierarchy for digital transmission
EZW	embedded zero-tree wavelet scheme
FFT	Fast-Fourier transform
FIR	finite impulse response
GHz	Giga Hertz
H(0)	zero-entropy
H(1)	first order entropy
H(2)	second order entropy

HTTP	Hypertext Transfer Protocol
HVS	human visual system
HWCS	hybrid wavelet-based compression system
ICT	irreversible colour transform
IDWT	inverse DWT
IFT	inverse Fourier transform
INS	Insignificant
ISDN	integrated signal digital network
IWT	inverse wavelet transform
IZ	isolated zero
JBIG	Joint Bi-level Image experts Group
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve transform
LIP	list of insignificant pixels
LIS	list of insignificant sets
LSP	list of significant pixels
MDCT	Modified DCT
MMS	Multimedia Messaging Service
Modem	modulator/demodulator
MPEG	Moving Pictures Experts Group
MRA	multiresolution analysis
MSE	mean square error
NEG	negative significant
NEZW	Naive-EZW
NSPIHT	Naive-SPIHT
PAL	phase alternating line
PCM	pulse code modulation
PDAs	personal digital assistants
PDF	portable document format
POS	positive significant
PSNR	peak signal-to-noise ratio
QMFs	quadrature mirror filters
r	coefficient of correlation

r^2	coefficient of determination
RGB	red, green, and blue
RMS	root mean square
RMS	root-mean-square
RMSE	RMS error
ROI	region of interest
ROIs	regions of interest
SBC	Subband Coding
SF	spatial frequency
SFM	spectral frequency measure
SIG	Significant
SNR	signal-to-ratio
SOI	subject of interest
SOT	spatial orientation tree
SP-CZW	Shapiro's dominant CZW
SPIHT	set partitioning in hierarchical trees
SQ	scalar quantization
Std. Dev.	standard deviation
STFT	Short Time Fourier Transform
SW	scalar wavelets
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	tagged image file format
UK	United Kingdom
VLC	variable length coding
VLSI	very large scale integrated
VQ	vector quantization
WMSE	weighted mean squared errors
YCrCb	Luminance (Y)/chrominance (Cr and Cb) colour space
YUV	Luminance (Y)/chrominance (U, V) colour space
ZRT	zero-tree root

Mathematical Notations

$\rightarrow$	approaches
Δ	the change in
$f(t)$	Function of continuous variable t
$F(w)$	Fourier transform
$\psi(t)$	wavelet function
$\langle , \rangle$	an inner product
$\int_{-\infty}^{\infty}$	integral with limits
$L^2(R)$	the vector space of all functions whose square are integrable
V^j	scaling space
$\phi(t)$	scaling function (mother wavelet)
$\in$	is an element of
$\langle$	is less than
$\rangle$	is greater than
$\lfloor a \rfloor$	largest number not greater than a
$\subset$	is a proper subset of
$h(n)$	set of coefficients
$\oplus$	direct sum
$\perp$	orthogonal
a_i	real-value expansion coefficients
$\sum$	the summation of
a_k^j	approximation coefficients
a_k	wavelet coefficients
Z	set of all integers
$\Leftrightarrow$	is logically equivalent to
W^j	detail space
$\{\}$	a set
$\ell^2(R)$	the vector space of all functions whose square are summable

$\hat{\phi}(\omega)$	Fourier transform of $\phi(t)$
$e^{j\omega t}$	complex exponential
$H(\omega)$	discrete Fourier transform of
$\Rightarrow$	implies
$\downarrow 2$	downsampling by a factor of 2
$\uparrow 2$	upsampling by a factor of 2
$x_n[n]$	digital signal signal
H_1	denotes low-pass filter (analysis filter bank)
H_2	denotes high-pass filter (analysis filter bank)
G_1	denotes low-pass filter (synthesis filter bank)
G_2	denotes high-pass filter (synthesis filter bank)
$A_{2^j}^d f$	discrete approximation signal
$D_{2^j} f$	discrete detail signal
d_k^j	detail coefficients
∞	infinity
$\cong$	defined as

CHAPTER 1

Introduction to the Research

1.1. Background

Over the past two decades visual communication has gained much popularity with mobile telephones and personal digital assistants (PDAs). Visual communication is also becoming increasingly important with applications in teleconferencing, telemedicine, digital television, entertainment, online learning and many more. Parallel to this growth of visual communication is the transmission of colour images that is, 'picture messaging' and the Multimedia Messaging Service (MMS). With MMS users can add images to text messages. Digital colour images are usually large and they require more storage and transmission time than grey-scale images. This can pose considerable problems for communication networks due to the potentially large volumes of data involved, for example, transmitting a 512 x 512 colour image coded using 24 bits (8 bits/pixel) via a modem at 28.8 kilobits/second would take about 3.6 minutes. Table 1.1 and Table 1.2 give examples of transmission times and storage capacity for uncompressed images [Oh and Woolley, 1999].

Table 1.1 Transmission time for uncompressed PAL video image [source: Oh and Woolley, 1999]

Duration of PAL video	File size (Kbits)	Transmission Time (hour)			
		Modem (56 Kbps)	ISDN2 (128 Kbps)	ISDN6 (384 Kbps)	E1 (2.048 Mbps)
1 sec	61440	0.3	0.1	0.04	0.008
2 min	7372800	36.5	16.0	5.3	1.0
5 min	18432000	91.2	40.0	13.3	2.5

Table 1.2 Digital media storage capacities for PAL video image[source: Oh and Woolley, 1999]

Storage Media	Maximum storage capacity	
	No of Frames	Duration of video (min)
Magneto-Optical (MO) – 640 Mbytes	2560	1.4
MO – 2.6 Gbytes	10649	5.9
Recordable-CD –680 Mbytes	2720	1.5
DVD – 8.5 Gbytes	34816	19.3

The data from the tables show that transmitting such a large number of bits using a normal communications link is very slow and the storage capacity is inefficient.

Over the years, the problem of limited bandwidth of transmission channels has been lessened by significant improvement in storage and transmission technologies. For example a Digital Versatile Disk (DVD) is capable of storing Gigabytes of movies and fibre optics can carry Terabits/sec. Broadband Internet access is available via Digital Subscriber Line (DSL) and Asymmetric Digital Subscriber Line (ADSL) technology. At the same time, there have been significant advances in image compression techniques (for example, JPEG and MPEG). This poses the question, is it worth continuing with research into image compression?

Despite the improvements made in media storage technology, compression techniques and the performance of transmission media as mentioned earlier, the demand for greater data storage capacity and faster transmission speeds will continue to exceed the capabilities of current technologies. Furthermore, there is still one application field that justifies putting efforts into image compression research. This application is mobile computing. Unlike their desktop counterparts, Internet access via mobile devices is badly affected by low air bandwidth. The radio spectrum is a scarce resource and it cannot be altered. The charts in Figure 1.1a and Figure 1.1b highlight the broad allocation of spectrum in the UK by various categories of use. This data was provided by the Radio Communications Agency and used by Cave [2002] in work based on an analysis of the detailed spectrum allocation table for the UK. It should be noted the information is based on an allocation table data rather than on licensing records, thus there may be some disparities between the amount of spectrum shown and the amount of spectrum actually issued under licence [Cave, 2000, pp.207].

Further to the earlier discussion, in recent years third generation (3G) mobile communications, broadband Internet access, and digital broadcasting delivered by terrestrial wireless or satellite have been deployed, which will invariably place additional demands on the radio spectrum. Therefore, radio spectrum management continuous to be a more important issue and has prompted the British government to review the way radio spectrum was being managed [Cave, 2002].

Figure 1.1a 88MHz – 1GHz allocation [source: Cave, 2000, pp.207]

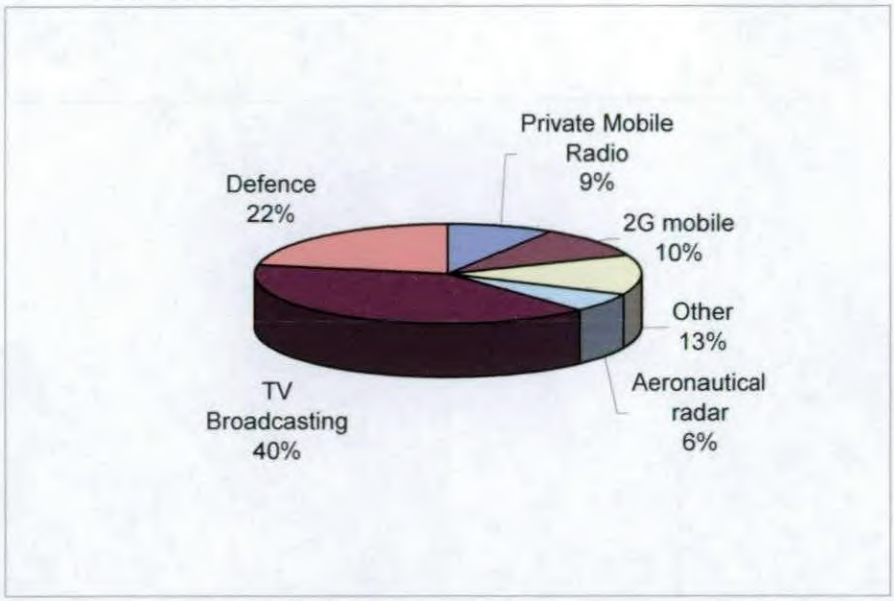
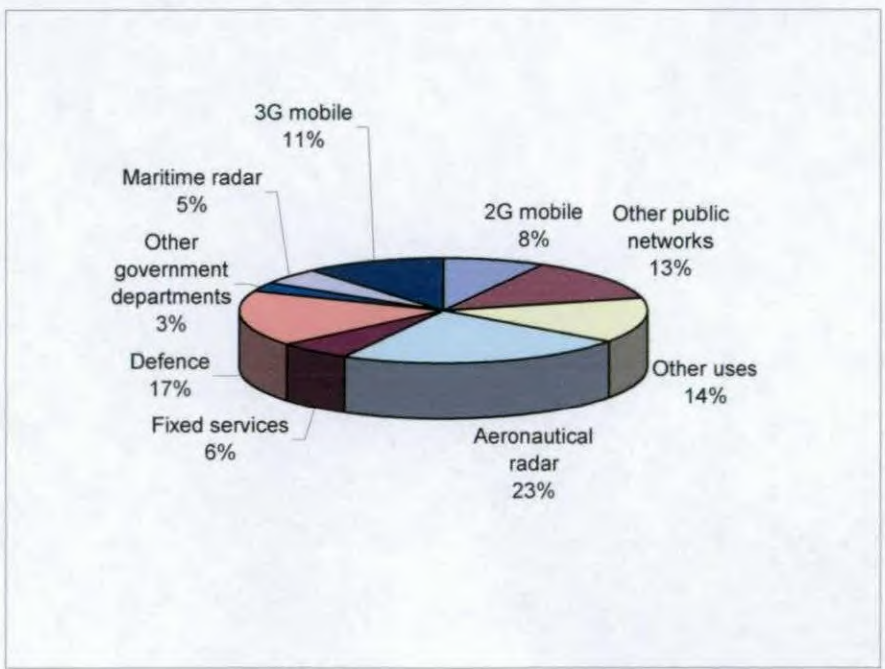


Figure 1.1b 1GHz – 3GHz allocation [source: Cave, 2000, pp.207]



The air bandwidth will remain a bottleneck, as the growth of Internet traffic using air bandwidth is likely to continue, as the number of mobile devices used to access the Internet is likely to increase. Furthermore, Internet protocols like TCP/IP and HTTP, come with a lot of overheads. These overheads require many messages between clients and server when

setting up a connection. These overheads render TCP/IP and HTTP unsuitable for wireless telecommunication [Schiller, 2000, pp.316-317]. Advances in communications channels (example 3G communication network) go some way to addressing this problem but the rapid increase in traffic and demand for ever better quality images, means that effective data compression techniques are essential for transmitting and storing digital images. The main objective of this thesis is to offer a novel image compression technique that can help to overcome the bandwidth problem.

1.2. Image Compression and Compression Standards

The previous section shows that effective data compression techniques are essential for transmitting and storing digital images and thus, image compression has become very important with the rapid growth in multimedia computing and the advent of the Internet. Gonzalez and Woods [1993, pp.308] regard image compression as an enabling technology and it is the subject of much research in industry and universities.

One of the most widely used image compression standards is the Joint Photographic Experts Group (JPEG) and has been the preferred method in still image compression for the past decade. JPEG is a discrete cosine transform (DCT) based compression standard that works best on 'continuous tone' or natural images but synthetic images, for example clipart, with many sudden jumps in colour values will not so compress well. Nevertheless a lot of the synthetic images used on the Internet are in JPEG format.

The JPEG compression algorithm first converts the original image into an YCrCb image. The image is then divided into 8×8 blocks and each block is transformed using the DCT. Compression is achieved via quantization followed by variable length coding (VLC) [Wallace, 1992]. Each step contributes to the overall compression of the image.

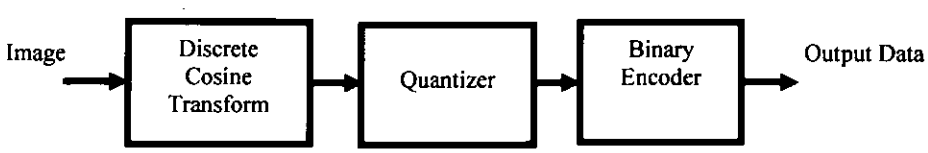


Figure 1.2 JPEG algorithm

JPEG has a fundamental limitation that derives from the block-based segmentation of the source image. Because of the block-based segmentation approach, a JPEG image can suffer from annoying blocking artefacts. JPEG is a very efficient coding method but the performance degrades at high compression ratios [Rao and Bopardikar, 1998, pp.167].

As mentioned earlier, JPEG is intended for continuous tone images. Other compression standards like JBIG also exist. JBIG is a lossless method for compressing bi-level or bi-tonal (black and white) [Jbig, 2002] images like regular printed text, for transmission across a communication channel, for example, facsimile. JBIG is not widely applied because there is no commonly used data format and there is no official JBIG file format. However, JPEG and JBIG are part of other standards such as the TIFF file format and page description languages like PDF.

In recent years researchers have focused on the utilization of discrete wavelet transforms (DWT) in digital image compression. The success of the DWT as a compression technique has prompted its inclusion in the JPEG2000 standard. The advantage of the DWT is that, in contrast to the DCT, it does not divide the image into blocks, but analyses the image as a whole. The wavelet transform allows a high compression ratio and yet maintains the image quality [DeVore et al, 1992].

However, there exists a large selection of wavelet families and researchers are faced with the task of choosing the most suitable wavelet for a particular application. Firstly, wavelets can be divided into continuous and discrete transform groups. Furthermore, discrete wavelet transforms can be orthogonal or biorthogonal. Which is the 'best' wavelet for coding colour images? Hubbard [1998, pp.239] cited that Frage caution against spending too much energy and that choosing a wavelet Choosing the best wavelet is quite a task. Hubbard [1998, pp.240] also cited that Rioul disagrees with Frage and asks, "*...why not try to find the wavelet best adapted to a given task...*" and even suggests custom-making "*...one's own wavelets, with the properties one wants...*".

A brief examination of a typical wavelet-based image coder system reveals that the encoder consists of three major parts, that is, the wavelet transform, a quantizer, and an entropy coder. The image is first decomposed into wavelet coefficients. The quantizer then quantizes the wavelet coefficients. The entropy coder produces an output bit stream and

then encodes these wavelet coefficients. Although the overall performance of the coder depends on all three parts, the choice of the wavelet will ultimately affect the performance of the coder. From the above discussion this thesis concurs that the search for the best wavelet for coding colour images is important.

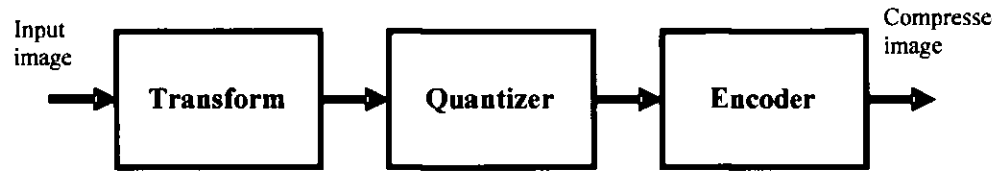
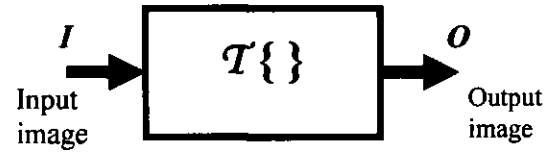


Figure 1.3 A typical image compression system

Wavelets have been used increasingly in compression systems. In spite of the widespread interest in the application of wavelets in image compression, limited studies have been conducted into the selection of the 'best' wavelet from a large family of wavelets available for a specific image.

In general, the following can represent the operation of a generic image coder;



$$O = T\{I\}$$

Figure 1.4 General representation of image coder

$T\{\}$ is the system transformation that maps the input image 'I' into the output image, 'O'. Therefore it can be said that the output image, 'O' is related to the input image, 'I'. If the output 'O' is related to the input image 'I', this could influence the performance of the coder. Applying this to a wavelet-based image compression system, the wavelet filter and the input image are important parameters and therefore warrant further investigation.

In the past, most research in image compression primarily dealt with grey-scale images and therefore most test images used are grey-scale. This is not realistic in the current environment where colour is a more attractive option. Hence this research is concerned

with compression of colour images rather than grey-scale images. The major concern of this research is to identify the best wavelet for compressing still colour images. Therefore, the initial research questions are:

1. What image statistics affect the coding performance?
2. Which wavelet filter is best for compressing a particular image?

Based on these questions, a systematic study of the effects of image statistics on coding performance was carried out.

1.3. Applications

There are two specific application of interest in this research, namely mobile computing and medical imaging. As mentioned earlier, 'picture messaging' and MMS are becoming popular but transmitting such a large amount of bits using a normal communications link can be slow and may be costly using mobile devices. Current mobile devices like PDAs have several constraints that is, limited processing, low display resolution, limited storage capacity, and relatively limited communication speed. While technological advances will reduce some of these constraints, mobile devices are likely to remain significantly less capable than their desktop counter parts. For example, resolutions on PDAs are limited by the compactness of the screens and a typical handheld PDA screen is about 4 inches diagonally, supporting 320 x 320 resolution. It is expected that screen resolution will continue to improve but screen sizes are likely to remain small because most mobile device users prefer compactness. Although the small screen of mobile devices is a disadvantage, it offers an opportunity for a trade-off between image quality and transmission speed. Another possible solution to the small screen factor is a scalable compression system in which the user can determine the level of image quality. This is possible with a wavelet-based compression system that can decompose the image to any desired level (in theory) and hence offer a trade off between the image quality (PSNR) and image size (bytes transmitted).

The other application area of interest in this research is in the compression of medical images. The usage of mobile devices like PDAs is becoming more widespread among doctors who need to access data such as medical images from remote locations [Ratib et al, 2003, Lowes, 2002]. As mentioned earlier, the bandwidth and storage constraints of mobile devices means that medical images too must be compressed before transmission and storage. The trade off between bit rate and image quality is much more important when

medical applications are being considered. Under some circumstances, speed and hence low bpp may be the important factor whereas in others, image quality will be paramount.

For both of the application areas of interest, the variability in the capability of the end user's mobile device is significant. A growing range of products is becoming available with some wide variations in the key features already identified; display size and resolution, processing power and storage capability. What is envisaged is a compression system that will be able to select an optimal compression method to meet both the demands specified by the user that is matched to their equipment. There is little point, for example, in transmitting a high resolution image to a device that only has a low resolution screen, even if the user has asked for it!

1.4. Overview of Research

As a major concern of this research is to identify the best wavelet for compressing still colour images, different wavelets were used to compress a selected set of colour images. The initial results show that different wavelets produce different varied coding performance. Coding performance refers to peak signal-to-noise ratio (PSNR), compression ratio (CR) and bits-per-pixel (bpp).

To explore and to understand how the various image features affect the coding performance, grey-scale image histogram features are used to analyse colour images. Grey-scale image feature analysis is being employed initially as there is limited data on coding performance measures for colour images. This will allow certain quantitative and qualitative comparisons to be made. The histogram features are statistically based features and they can provide information about the characteristics of the colour distribution of a colour image. These features include first order statistics, for example mean, standard deviation, variance, energy, entropy, skewness, kurtosis, and entropy. Other characteristics explored include, image gradient. Frequency characteristics of the images such as spectral frequency measure (SFM) and spatial frequency (SF) are also explored.

Different approaches to compressing colour images using wavelets are also explored. One of the methods explored is the utilization of a dual wavelet to compress colour images. The proposed dual wavelet compression scheme uses one wavelet to compress the luminance image and another to compress the chrominance images. The results, although confined to

a small sample of images and range of wavelets, clearly show that potential gains can be achieved by the application of dual wavelets. The work on the dual wavelet scheme also showed that coding performance could be further improved with a single wavelet by processing the luminance and chrominance images at different levels of decomposition.

Another approach explored is a wavelet-based image compression scheme in which a grey-scaled image is first partitioned into non-smooth and smooth segments and different wavelets are then applied using the 2-D DWT. Initial results indicate that some improvement in compression can be achieved through the use of a combination of wavelets with little loss in the quality of the reconstructed image. This could prove to be a very significant factor for mobile devices, where bandwidth costs can be high and small reductions in PSNR would have little, if any, impact. The approach is extended to colour images. The method first partitions the luminance image into smooth and non-smooth segments and different wavelets are then applied using the 2-D DWT. The results show that the combination of segmentation and wavelet selection based on image properties can produce some improvements in compression.

The final approach explored is an image compression system called a hybrid wavelet-based compression system (HWCS), which is based on the assumption that viewers are more interested in the subject of interest (SOI) for a given image than the background. A subjective evaluation of the visual quality of the images indicates that this approach is broadly successful.

1.5. Organization of Thesis

The following chapters of this thesis are organised as follows.

Chapter 2 presents the mathematical background of transforms used in the field of compression with a particular emphasis on transforms used in image coding. The reason for this is that, the researcher views the transform of an image coder as an important focus in this research. Three major transforms are reviewed in detail, that is, the Fourier transform, continuous wavelet transform and discrete wavelet transform.

Chapter 3 is a literature survey and review of the work done in the field of image coding, with a particular emphasis on subband coding and wavelets. Mallat's multiresolution analysis, which is the core of most wavelet-based compression schemes, is also examined in this chapter. The literature review explores work using DCT based image compression methods, in particular the JPEG standard and its problems. The JPEG2000 standard, a wavelet-based compression scheme, is also discussed in detail.

Chapter 4 is an extension of chapter 3 and begins with a consideration of early wavelet-based compression schemes. These early schemes can be traced back to the works of Queiroz et al's [1997] and Huh and Hwang's [1997]. Advanced wavelet-based compression schemes, that use novel quantization and encoding techniques, like embedded zero-tree wavelet scheme (EZW) [Shapiro, 1993] and SPIHT [Said and Pearlman, 1996] are explored in detail. The chapter also explores wavelet-based compression schemes applied to colour images.

Chapter 5 begins with a look at two common colour spaces, that is the RGB and YCbCr colour spaces. A study of how different wavelet filters affect the coding performance of colour images was carried out and the results are reported. Also included in this chapter is a detailed study of which image histogram statistics have a connection with the coding performance measures.

Chapter 6 explores different approaches to compressing colour images using wavelets. This chapter begins by describing a method for identifying the frequency characteristics of a colour image. Next, a dual wavelet compression scheme to compress colour images is described. The scheme compresses colour images using different wavelet filters for the luminance component and chrominance components. Section 6.4 describes a segmentation-based wavelet image compression scheme to compress grey-scale and colour images. In this scheme the image is first partitioned into non-smooth and smooth segments and different wavelets are then used to compress them.

Chapter 7 starts with a look at the concept of 'subject of interest' (SOI) used in photography. This chapter describes in detail an image compression system based on the subject of interest called a hybrid wavelet-based compression system (HWCS). The HWCS

is based on the assumption that viewers are more interested in the subject of interest for a given image than the background.

Chapter 8 summarizes the main results of this thesis and puts forward ideas for future work.

CHAPTER 2

Mathematical Background – *From Fourier to Wavelet transform*

2.1. Introduction

One of the key components in image processing is the transform. Several transforms have been used for image compression and these include Karhunen-Loeve, lapped orthogonal, discrete cosine, and more recently, wavelets. As one of the main concerns of this research is image compression this section will therefore examine three relevant transforms, the Fourier transform, continuous wavelet transform, and discrete wavelet transform (DWT). The reason for examining these three transforms is that they are essential in the understanding of the work of this research and it will be shown in this chapter that the mathematics of wavelets can be traced back to the Fourier series and Fourier transform. Included in this section is also a review of filter banks and their role in image compression.

As mentioned earlier, wavelet mathematics underpinnings can be traced back to the Fourier series and Fourier transform. Therefore, it will be logical and beneficial to begin this chapter with a look at Fourier series and the Fourier transform.

2.2. Fourier series

Jean Baptiste Joseph Fourier introduced the concept that any function can be expressed as a series of sinusoidal waves that are multiples of a basic frequency [Bolton, 1995, pp.12] and this can be expressed mathematically as;

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{n=\infty} a_n \cos nwt + \sum_{n=1}^{n=\infty} b_n \sin nwt \qquad (2.1)$$

Equation (2.1) is the Fourier series in trigonometric form.

2.3. Fourier Transform

Aperiodic signals can be represented as weighted integrals of complex sinusoids that are not harmonically related [Qian, 2002, pp.32]. The Fourier series can be extended to an aperiodic because aperiodic signals can be viewed as periodic signal with an infinitely long period, which can be derived by considering the exponential form of the Fourier series.

$$f(t) = \sum_{n=-\infty}^{n=\infty} c_n e^{jnwt} \tag{2.2}$$

This exponential form of the Fourier series can be rewritten

$$f(t) = \sum_{n=-\infty}^{n=\infty} c_n e^{\left(jn\frac{2\pi}{T}t\right)} \text{ where } w = \frac{2\pi}{T}. \tag{2.3}$$

As the period T increases, the fundamental frequency $\frac{2\pi}{T}$ decreases, and the harmonically related components become closer in frequency.

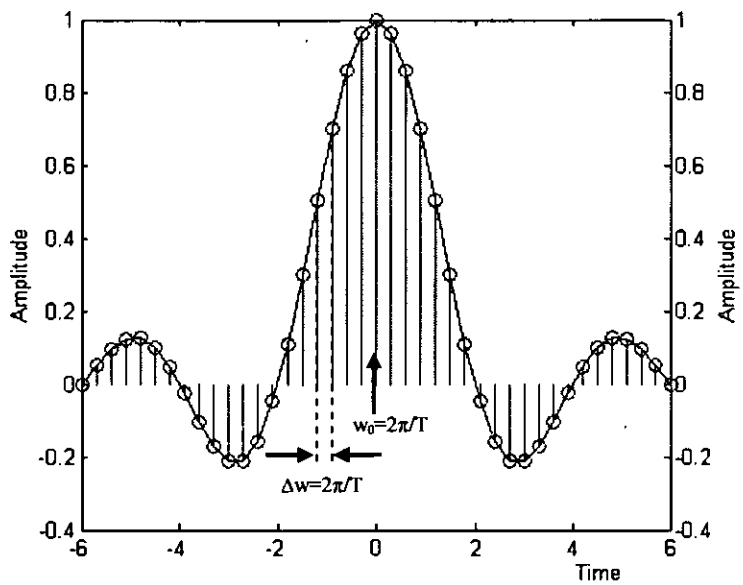


Figure 2.1 Sinc wave

As $T \to \infty$ the spacing between harmonics, Δw, becomes smaller and smaller, and the amplitude spectrum becomes a continuous graph [Balmer, 1998, pp.232-234].

With very small Δw intervals, the discrete sum becomes a continuous sum and so the summation becomes an integral. Thus, the Fourier series equation becomes

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)\, e^{jwt}\, dw \tag{2.4}$$

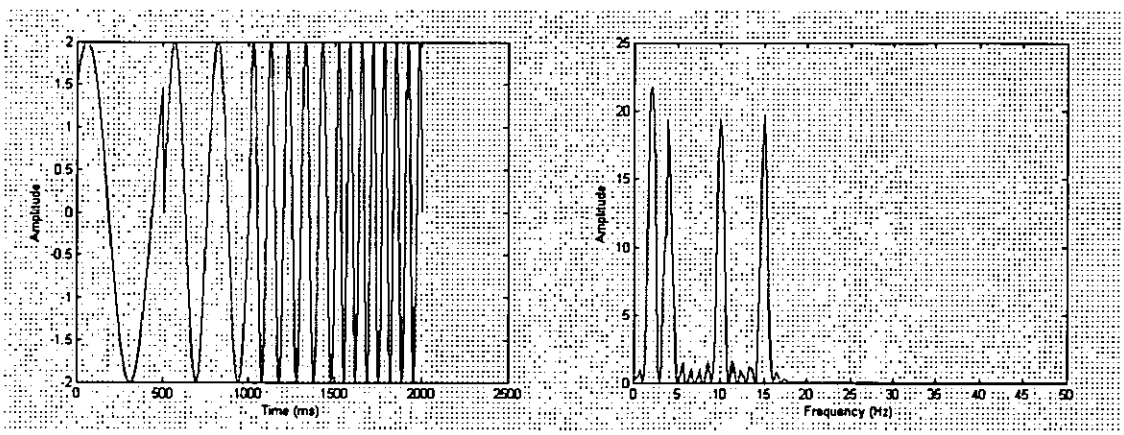
where

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-jwt}\, dt \tag{2.5}$$

$F(w)$ is the measure of the similarity between the signal $f(t)$ and complex sinusoidal functions. Equation 2.5 is known as the continuous Fourier transform and equation 2.4 is the inverse Fourier transform (IFT). In the frequency domain, the function or signal $f(t)$ can be analysed for its frequency contents. Another advantage is that the signal $f(t)$ can also be filtered to improve or eliminate certain frequency components. The IFT re-transforms the processed signal $F(w)$ back into the time domain.

The Fourier transform is a powerful and common mathematical tool for signal processing. However, it has drawbacks, notably it can be computed for only a single frequency at any one time. Secondly, the Fourier transform can only provide information in the frequency domain and no information in the time domain. In other words, time information is lost [Misiti, 1997, pp.1-3]. The loss of time is due to the integration over time from $-\infty$ to $+\infty$, hence time locality disappears.

If the signal to be analysed is non-stationary and contains some kind of discontinuity, then the Fourier transform does not provide any information on where the discontinuity is in time. Figure 2.2a shows a non-stationary signal with frequency components of 2 Hz, 4 Hz, 10 Hz, and 15 Hz. There is a small discontinuity between the 2 Hz and 4 Hz components. However, the spectrum shows no indication of the discontinuity.



(a) (b)

Figure 2.2(a) Signal with a discontinuity and (b) its Fourier transform

2.4. Short Time Fourier Transform

The Short Time Fourier Transform (STFT), which is also known as the 'Windowed Fourier Transform' [Phillips, 2003; Goswami and Chan, 1999, pp.60], was an attempt by Dennis Gabor to overcome the lack of information on time locality in Fourier analysis [Gabor, 1946]. Gabor modified the Fourier transform to analyse non-stationary signals by dividing the signals into small sections. These small sections of the signal can then be assumed to be stationary and their Fourier representation can then be computed. The result is a signal that is mapped into a two-dimensional function of time and frequency.

Recall the Fourier transform is defined as;

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-jwt}\, dt$$

and the STFT is defined by;

$$STFT(\tau, f) = \int_{-\infty}^{\infty} f(t) g^{*}(t-\tau) e^{-jwt} \qquad (2.6)$$

The difference between the Fourier transform equation and the STFT is the function $g^{*}(t-\tau)$ [Rioul and Vetterli, 1991]. The '*' indicates a complex conjugation.

The function $g^{*}(t-\tau)$ can be thought of as a window that is shifted along the signal $f(t)$. For each shift, the Fourier transform of the product function $f(t)g^{*}(t-\tau)$ is computed. In the following figure, a window function is placed at t = -3. The window function and the signal function $f(t)$ are then multiplied. The Fourier transform of the product can then be computed. The window is then shifted to a new position, multiplied with the signal $f(t)$, and the Fourier transform of the product is then computed. The procedure is repeated until the end of the signal is reached.

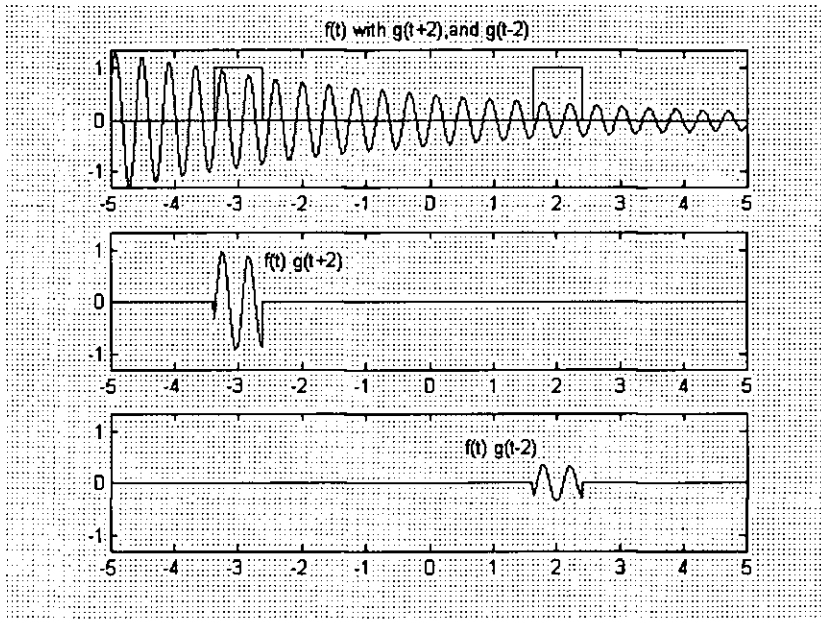


Figure 2.3 Computing STFT

The variables $(t-\tau)$ are the transform variables from the single time domain variable *t*. Therefore, it could be said that the STFT transforms a time-variable function (*t*) into a time-frequency function.

The signal *f(t)* may then be reconstructed using:

$$f(t)g^*(t-\tau)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(w)e^{-jwt} \qquad (2.7)$$

The STFT does provide some information on both time locality and frequency spectra, but it does have a limitation. The limitation is that once a particular window size is chosen the same window is used to analyze every frequency of the signal. A small window results in good time resolution but poor frequency resolution. A large window gives good frequency resolution but poor time resolution.

To illustrate the effects of different window size, the STFT is applied to a non-stationary signal (Figure 2.4) that consists of four sinusoids of 2Hz, 4Hz, 7Hz, and 9Hz. For the first example, the window used is a Hamming window with a width of 100mS. The STFT of the signal is shown in Figure 2.5a.

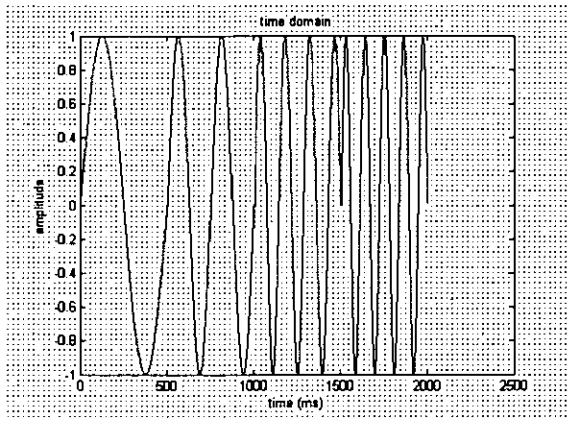
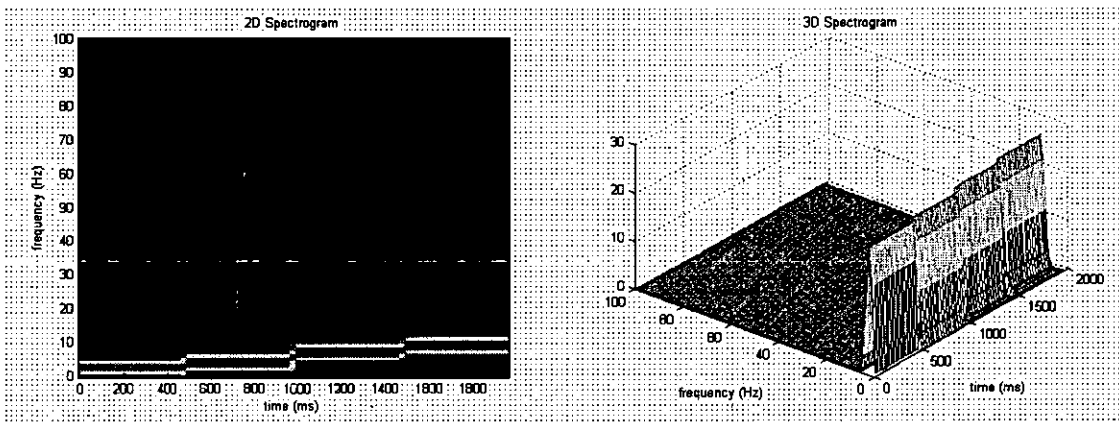


Figure 2.4 Signal in the time domain

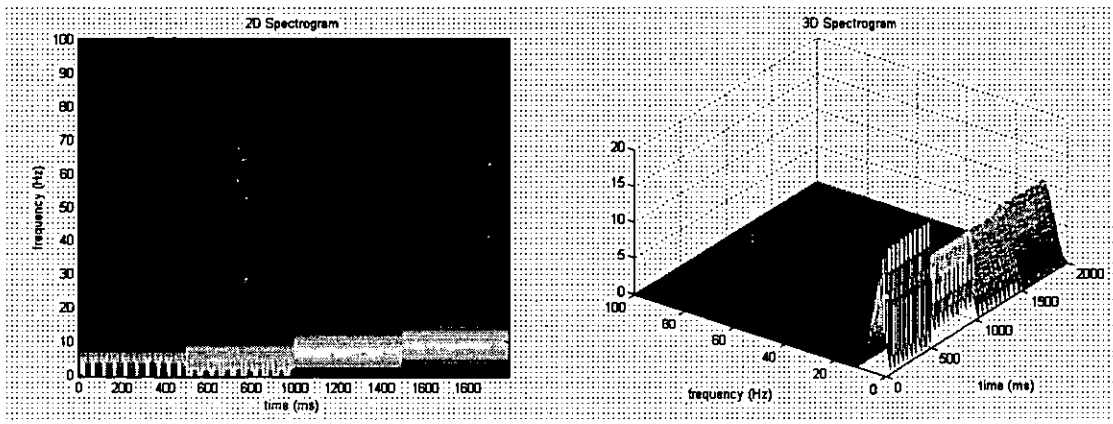
Figure 2.5a is a spectrogram that shows frequency against time. This figure indicates that frequencies can be resolved but not time when the window width is 100ms.



(a) (b)

Figure 2.5 (a) STFT of signal (b) 3D spectrogram of (a)

Figure 2.5b shows the 3D spectrogram of Figure 2.5a. The figure shows four frequency components at different time intervals. With a window width of 40mS the following plots are obtained.



(a) (b)

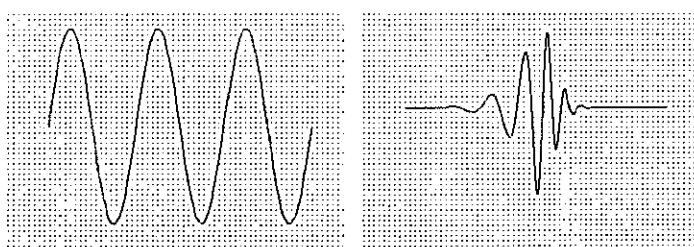
Figure 2.6 (a)STFT of signal with narrower window width (b)3D spectrogram of (a)

With a narrower window, the frequencies cannot be resolved but time is resolved. The reason for this limitation may be reduced to the Heisenberg Uncertainty Principle [Rao and Bopardikar, 1998, pp.18-19].

2.5. Wavelets

The continuous wavelet transform (CWT) is an alternative approach to the STFT. The CWT is very similar to the STFT in that the signal is multiplied with a function and the transform is then computed for different parts of the time-domain signal.

A wavelet is a waveform that is limited in duration and has an average value of zero [Misiti, 1997]. Wavelets tend to be irregular and asymmetric. On the other hand, sinusoids are smooth, predictable and they do not have limited duration. See Figure 2.7.



(a) (b)

Figure 2.7 (a) Sine wave (b) wavelet

To understand the CWT, it is essential to examine two important ideas, that is, translation and scaling. These ideas will be discussed in the following sub-section.

2.5.1. Translation and Scaling

For the purpose of discussion, a specific function will be used. Consider the function $f(t)$, where:

$$f(t) = \cos(5t)e^{-x^2} \tag{2.8}$$

Equation 2.8 describes a cosine function multiplied by an exponential. The result is a function that decays rapidly.

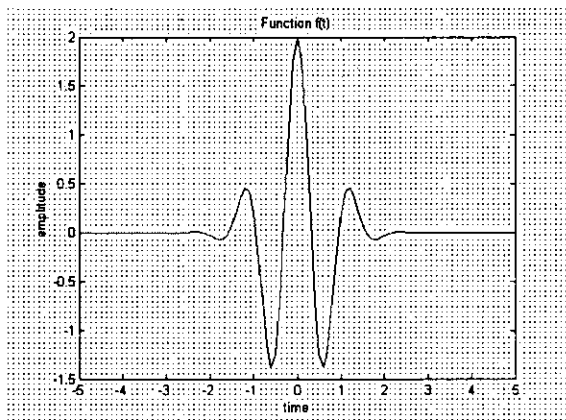


Figure 2.8 $f(t) = \cos(5t)e^{-x^2}$ *plot*

Assuming that $f(t) = \cos(5t)e^{-x^2}$ is a wavelet, then a function to be decomposed is expressed as a linear combination of different translations and scalings of this wavelet.

Translation can be achieved by subtracting a value τ from the variable t, that is, $f(t-\tau)$. This translates the function $f(t)$ by τ units and the equation is written as $f_t(t) = \cos(5t-\tau)e^{-x^2}$. Figure 2.9(a) shows how the function $f(t) = \cos(5t)e^{-x^2}$ is translated or shifted by 6 units along the x-axis.

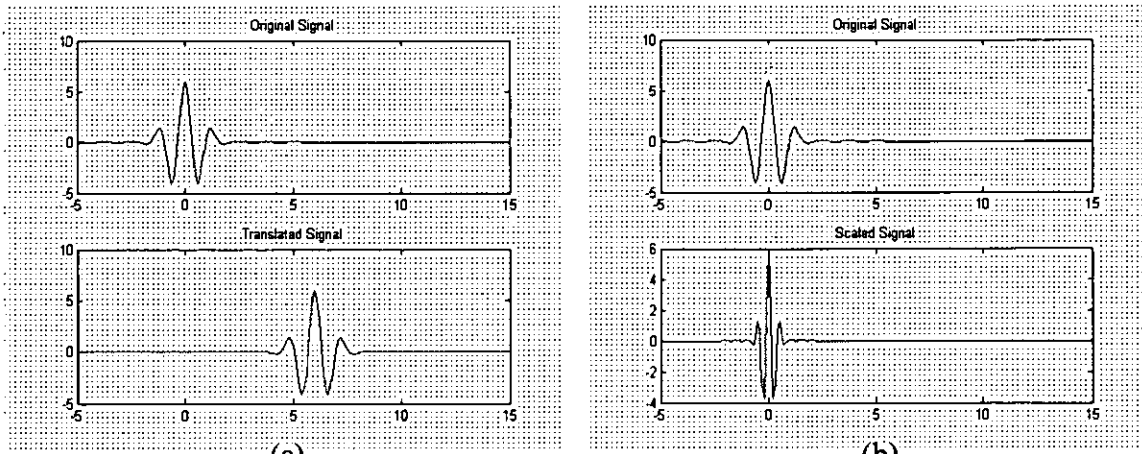


Figure 2.9 (a) Translated and (b) scaled signal

If the variable t is divided by a scaling factor s the function $f(t)$ will be stretched or compressed. Figure 2.9(b) shows how the function $f(t)=\cos(5t)e^{-t^2}$ is scaled by a factor of 0.4. The function is stretched or compressed, depending on the scaling factor 's'. The scaling factor 's' may be greater than one or less than one. The scaled and translated wavelet can be written as:

$$w_s^{\tau}(t)=f\left(\frac{(t-\tau)}{s}\right) \tag{2.9}$$

where s and τ refer to a scaling and translation of the wavelet $f(t)$.

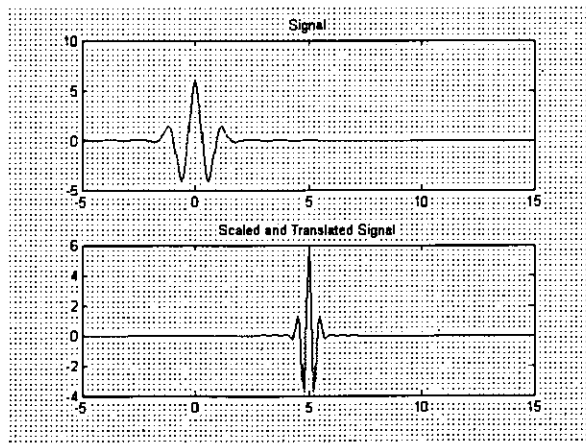


Figure 2.10 Scaling and translating

2.6. The Continuous Wavelet Transform

Any function $\psi(t)$ is a 'mother wavelet' or wavelet if it satisfies the following two properties [Rao and Bopardikar, 1998, pp.1-2]:

1. The function integrates to zero:

$$\int_{-\infty}^{\infty}\psi(t)dt = 0 \tag{2.10}$$

2. The function is a square integral or, equivalently, has finite energy:

$$\int_{-\infty}^{\infty}|\psi(t)|^2\, dt < \infty \tag{2.11}$$

Let $f(t)$ be any square integral function. The CWT of $f(t)$ with respect to a wavelet $\psi(t)$ is defined as;

$$CWT_f(\tau,s) \cong \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{|s|}}\psi^*\left(\frac{t-\tau}{s}\right)dt \tag{2.12}$$

where '$\cong$' stands for '*is defined as*'.

'τ' represents the time shift or translation and 's' determines the amount of scaling, it is also known as the scale and or dilation variable. $\psi^*\left(\frac{t-\tau}{s}\right)$ is the transforming function or the 'mother wavelet', where '*' indicates conjugation. This mother wavelet is similar to equation (2.9). The wavelet transform is a function of two variables 'τ' and 's'. The $\frac{1}{\sqrt{|s|}}$ factor is known as the normalising factor which ensures that the energy stays the same for all of 'τ' and 's'.

Equation (2.12) can be rewritten in a more compact form by defining $\psi_s^\tau(t)$ as

$$\psi_s^\tau(t) = \frac{1}{\sqrt{|s|}}\psi\left(\frac{t-\tau}{s}\right) \tag{2.13}$$

Combining equations (2.12) and (2.13) gives,

$$CWT_f(\tau,s) = \int_{-\infty}^{\infty} f(t)\psi_{\tau,s}^*(t)dt \tag{2.14}$$

The equation (2.14) can be seen as an inner product that is

$$\langle f(t), \psi_s^{\tau}(t) \rangle = \int_{-\infty}^{\infty} f(t) \psi_{\tau,s}^{*}(t) dt . \qquad (2.15)$$

The equation (2.15) represents the inner product of two functions, defined in the $L^2(R)$ space. The CWT computes, via the inner product formula, the wavelet coefficient of $f(t)$ associated with the wavelet $\psi_s^{\tau}(t)$ [Liu and Chan, 1998, pp.7]. The coefficient indicates the correlation of the signal $f(t)$ with the scaled and translated wavelet $\psi_{\tau,s}^{*}(t)$. The correlation is a measure of the 'similarity' between the signal $f(t)$ and the wavelet $\psi_{\tau,s}^{*}(t)$. This view is illustrated in Figure 2.15.

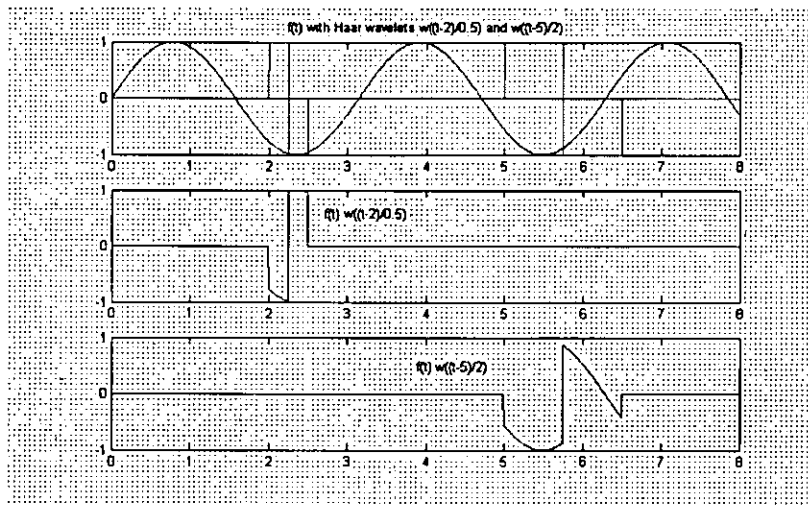


Figure 2.15 f(t) along with `haar` *wavelet*

The Figure 2.15 shows a signal $f(t)$ along with the `haar` wavelet with two different translations and scales. For illustration purposes the `haar` wavelet is used because it the simplest of all the wavelets.

2.7. Discrete Wavelet Transform

With reference to the above figure (Figure 2.15), it can be seen that in order to capture the full characteristics of the signal $f(t)$, the coefficients have to be calculated for the whole of the signal. The computation load can be quite heavy depending on the size of the window. To circumvent this problem, the discrete wavelet transform (DWT) was developed. The DWT algorithm is identical to that of a two-channel filter bank analysis.

The original function $f(t)$ can be reconstructed by:

$$f(t)=\frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{s^2}CWT_f(\tau,s)\psi_{\tau,s}\left(\frac{t-\tau}{s}\right)dsd\tau \tag{2.16}$$

where C_ψ is a constant that depends on the choice of wavelet and is given by

$$C_\psi = \int_{-\infty}^{\infty}\frac{|\Psi(\omega)|^2}{|\omega|}d\omega \tag{2.17}$$

where $\Psi(\omega)$is the Fourier transform of the mother wavelet $\psi(t)$. Equation 2.17 is known as the admissibility condition [Vetterli and Kovacevic, 1995, pp.301]. The admissibility condition implies $\Psi(0)=0$, that is:

$$\Psi(0)=\int_{-\infty}^{\Re}\psi(t)dt=0 \tag{2.18}$$

which means the mother wavelet $\psi(t)$ has to be a bandpass filter in the frequency domain. To recover from its inverse wavelet transform (IWT), $\psi(t)$ must satisfy equation 2.18. This condition restricts the selection of functions, that is, mother wavelet, $\psi(t)$. As far as the CWT is concerned, practically any function can be called a wavelet, provided it has a zero integral.

The CWT generates a lot of redundant data. Alternatively, the original signal or function can be reconstructed by a sampled version of $CWT_f(\tau,s)$. The $CWT_f(\tau,s)$ .can be sampled in a dyadic grid, that is, $\tau=k2^{-j}$ and $s=2^{-j}$ [Goswami and Chan, 1999, pp.72; Vetterli and Herley, 1992]. Substituting τ and s into equation 2.12 gives

$$CWT(k2^{-j},2^{-j})=\int_{-\infty}^{\infty}f(t)\frac{1}{\sqrt{|2^{-j}|}}\psi^*\left(\frac{t-k2^{-j}}{2^{-j}}\right)dt$$

$$=\int_{-\infty}^{\infty}f(t)\sqrt{2^j}\psi(2^jt-k)$$

$$CWT_f(k2^{-j},2^{-j})=\int_{-\infty}^{\infty}f(t)\psi_{j,k}^*(t)dt=a_k^j \tag{2.19}$$

where $\psi_{j,k}(t)$ is the dilated and translated mother wavelet $\psi(t)$ defined by

$$\psi_{j,k}(t)=2^{j/2}\psi(2^jt-k) \tag{2.20}$$

Alternatively, $\psi_{j,k}(t)$ can be written as

$$\psi_k^j(t)=2^{j/2}\psi(2^jt-k) \tag{2.21}$$

The original signal can be recovered from the sampled wavelet transform by

$$f(t)=\sum_{k}\sum_{j}a_k^j\psi_k^j(t) \tag{2.22}$$

where a_k^j is a two-dimensional set of coefficients known as the *discrete wavelet transform* (DWT) of $f(t)$. The double summation in equation 2.22 indicates that the wavelets have two parameters and they are translation and scale.

Functions or signals can be represented by a sum of scaling functions and/or wavelets. In other words, wavelets can be used to represent functions as a series expansion. A series expansion can be expressed as:

$$f(t)=\sum_{i}a_i\psi_i(t) \tag{2.23}$$

where i is an integer index of the finite or infinite sum, a_i is the real-value expansion coefficients, and $\psi_i(t)$ is the expansion set or basis set (basis function). The expansion set is a set of real-value functions of t. Wavelet decomposition is achieved with $\psi_i(t)$, which is also know as the 'mother wavelet' or generating wavelet. The expansion set or mother wavelet $\psi_i(t)$ should match the features of $f(t)$ in order to represent $f(t)$ using only a few coefficients. Therefore, the choice of $\psi_i(t)$ is important.

Wavelets can be orthogonal that is their 'inner products' are zero

$$\langle\psi_m(t),\psi_n(t)\rangle=\int\psi_m(t),\psi_n(t)dt=0 \tag{2.24}$$

The wavelet coefficients a_k can also be computed by the inner product

$$a_k=\langle f(t),\psi_k^j(t)\rangle=\int f(t)\psi_k^j(t)dt \tag{2.25}$$

2.8. Multiresolution Analysis

The function $f(t)$ can be decomposed into different scale levels, and each level can be further decomposed into different scale levels. This ability to decompose a function into different scale levels is called multiresolution analysis (MRA). Multiresolution is a mathematical tool that is often used in wavelet-based image compression schemes. In

multiresolution analysis two functions are required, that is a scaling function ϕ and a mother wavelet ψ.

2.8.1. The Scaling Function

The basic scaling function is defined as

$$\phi_k(t) = \phi(t-k) \qquad k \in Z,\ \phi \in L^2$$

where $\phi_k(t)$ is known as a *scaling function*, and Z is the set of all integers. A scaling function can be viewed as an element of a function space and not as a function that is defined by a formula. A function space is a collection of functions that can be represented by a sum of scaling functions and/or wavelets. One commonly used function space in signal processing is the $L^2(R)$ space. The $L^2(R)$ space contains all functions, which have a finite, well-defined integral of the square.

The set of functions or expansion set $\phi_k(t)$ can span (generate) a subspace of $L^2(R)$. This subspace $L^2(R)$ is defined as

$$V^0 = \overline{Span\{\phi_k(t)\}}$$

for all integers k from $-\infty$ to ∞. The bar indicates closure. This means that

$$f(t) = \sum_k a_k \phi_k(t) \qquad \text{for any } f(t) \in V^0.$$

A larger subspace can be spanned by changing the time scale of the scaling function. The basic scaling function can span a set of 2D functions by scaling and translation, which may be defined by

$$\phi_k^j(t) = 2^{j/2}\phi(2^j t - k) \tag{2.26}$$

The subspace $L^2(R)$ spanned is defined as

$$V^j = \overline{Span_k\{\phi_k(2^j)\}} = \overline{Span_k\{\phi_k^j(t)\}} \text{ for } k \in Z.$$

This means that if $f(t) \in V^j$, then it can be expressed as

$$f(t) = \sum_k a_k \phi(2^j t - k)$$

when $j > 0$, $\phi_k^j(t)$ is narrow and the span is large. This can therefore represent finer details. When $j < 0$, $\phi_k^j(t)$ is wider and is translated in larger steps [Burrus et al, 1998, pp.12]. In other words, wider scaling functions represent coarse information.

The function $f(t)$ is obtained as a linear combination of a dilation of $\phi(t)$ by a factor of 2^j and translated by k. In other words, the wavelet, $\phi_k^j(t)$ is compressed j times and shifted k times.

$$compressed: \phi_0^j = \phi(2^j t) \qquad shifted: \phi_k^0(t) = \phi(t-k)$$

In MRA the whole function space is decomposed into subspaces. This implies that when each function is decomposed, there will be a part of $f(t)$ in each subspace. The basic requirement of MRA is a nesting of spanned spaces:

$$\{0\} \subset ... V^{-2} \subset V^{-1} \subset V^0 \subset V^1 \subset ... \subset V^j \subset V^{j+1} ... \subset L^2$$

V^0 is known as the central space. Each V^j is contained in the next subspace V^{j+1}, that is, the subspace V^j is contained in all the higher subspaces.

The space spanned by the scaling function $\phi(2^j t - k)$ can be denoted by V^j and this is possible if $\phi(t) \in V^1$. This implies that if $\phi(t)$ is in V^0, it is also in V^1, which is spanned by $\phi(2t)$. $\phi(t)$ can be expressed, in terms of a weighted sum of shifted $\phi(2t)$, as:

$$\phi(t) = \sum_n h(n) 2^{1/2} \phi(2t-n), \qquad n \in Z \qquad (2.26)$$

where $h(n)$ is the scaling function coefficients and $2^{1/2}$ maintains the norm of the scaling function with the scale of two. This equation is known as the *recursive equation* or *dilation equation* [Burrus et al, 1998, pp.50; Goswami and Chan, 1999, pp.91].

2.8.2. The Wavelet Function

The wavelet equation can be represented by:

$$\psi(t) = \sum_n h(n) 2^{1/2} \phi(2t-n), \qquad n \in Z \qquad (2.27)$$

This equation produces wavelet functions that reside in the space generated by the scaling function $\phi(t)$. The equation is a weighted sum of shifted scaling function $\phi(2t)$ defined in the wavelet equation.

The wavelet equation generates the wavelet function, which gives the mother wavelet of the form:

$$\psi_k^j(t) = 2^{j/2}\psi(2^j t - k), \qquad j,k \in Z \tag{2.28}$$

where $2^{j/2}$ maintains the norm of the wavelet function of scale j. The set of functions $\psi_k^j(t)$ generate the differences between the spaces generated by the various scales of the scaling functions and these spaces can be denoted by W^j. W^j are known as the detail spaces and they are orthogonal to each other.

If the approximation $f(t)$ at level j is denoted by $f^j(t)$, then

$$f^j(t) \in V^j.$$

Since information at resolution level j is necessarily included in the information at a higher resolution, V^j must be contained in V^{j+1}, that is mathematically,

$$V^j \subset V^{j+1} \quad j \in Z.$$

This also implies that

$$f^{j+1}(t) \in V^{j+1}.$$

It can be deduced from earlier discussion that there will be a part of $f(t)$ in each subspace. In other words $f_j(t)$ is the part in V^j. A function in one subspace is in all higher (finer) subspace. Because of the definition of V^j, the spaces have to satisfy a natural scaling condition:

$$f(t) \in V^j \Leftrightarrow f(2t) \in V^{j+1}$$

which insures elements in a space are simply scaled versions of the elements in the next space [Burrus et al, 1998, pp.12].

The difference between $f^{j+1}(t)$ and $f^j(t)$ is denoted by

$$g^j(t) = f^{j+1}(t) - f^j(t) \tag{2.29}$$

then

$$f^{j+1}(t) = f^j(t) + d^j(t) \tag{2.30}$$

Consequently, the subspaces can be decomposed and written as

$$V^{j+1} = V^j \oplus W^j$$

where V^j is called the scaling space and W^j is called the detail space at resolution level j and is orthogonal to V^j. The $\oplus$ symbol indicates a direct sum of orthogonal subspaces.

$V^j \oplus W^j$ is the set of all elements $v^j + w^j$ where $v^j \in V^j$ and $w^j \in W^j$, and so $v^j \perp w^j$, which is used to denote v^j is orthogonal to w^j.

When the inner product between any element in W^j and any element in V^j is zero that is $\langle W^j, V^j \rangle = 0$, then it can be said that V^j is orthogonal to W^j. $V^j \perp W^j$ is used to denote V^j is orthogonal to W^j. This relationship can be described by the following diagram;

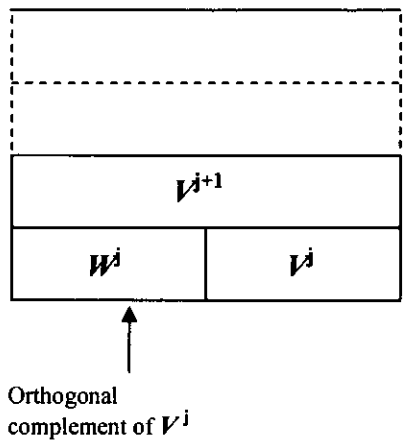


Figure 2.16 Wavelet subspaces

If the V space is further decomposed, then the following is obtained

$$\begin{aligned} V^{j+1} &= W^j \oplus V^j \\ &= W^j \oplus W^{j-1} \oplus V^{j+1} \\ &= W^j \oplus W^{j-1} \oplus W^{j-2} \oplus \ldots \end{aligned}$$

So the approximation space at resolution j, V^j, can be written as a sum of subspaces. These subspaces are mutually orthogonal [Liu and Chan, 1998, pp.10].

The subspaces $\{V^j\}$ are generated by a scaling function $\phi(t)$ and the subspaces $\{W^j\}$ are generated by a wavelet $\psi(t)$. Any function

$$f^j(t) \in V^j$$

can be written as

$$f^j(t) = \sum_{k=-\infty}^{\infty} a_k^j \phi(2^j t - k) = \sum_{k=-\infty}^{\infty} a_k^j \phi_k^j$$

and

$$g^j(t) \in W^j$$

can be written as

$$g^j(t) = \sum_{k=-\infty}^{\infty} d_k^j \psi(2^j t - k) = \sum_{k=-\infty}^{\infty} d_k^j \psi_k^j$$

where $\{a_k^j\}_{k\in Z}$ and $\{d_k^j\}_{k\in Z}$ are coefficients sets in $\ell^2(R)$.

2.9. The Frequency Domain

So far the discussion has been limited to the time domain. This section will examine the relations discussed in the previous section in the Fourier or frequency domain. The Fourier transform of the scaling function is represented by

$$\hat{\phi}(\omega) = \int_{-\infty}^{\infty} \phi(t) e^{-j\omega t} dt \tag{2.31}$$

where $\hat{\phi}(\omega)$ denotes the Fourier transform of $\phi(t)$, the scaling function. Recall the dilation equation is defined as

$$\phi(t) = \sqrt{2} \sum_k h_k \phi(2t - k) \tag{2.32}$$

If the dilation equation is multiplied by $e^{j\omega t}$ and integrated, then

$$\begin{aligned}
\int_{-\infty}^{\infty} \phi(t) e^{-j\omega t} dt &= \sum_k \sqrt{2} h_k \int_{-\infty}^{\infty} \phi(2t - k) e^{-j\omega t} dt \\
&= \left(\sum \frac{h_k}{\sqrt{2}} e^{-jk\omega/2} \right) \int_{-\infty}^{\infty} \phi(2t - k) e^{-j(2t-k)\omega/2} d(2t) \\
&= \left(\sum \frac{h_k}{\sqrt{2}} e^{-jk\omega/2} \right) \hat{\phi}\left(\frac{\omega}{2} \right)
\end{aligned}$$

Let $H(\omega) = \sum_k \frac{h_k}{\sqrt{2}} e^{-jk\omega}$ then

$$\hat{\phi}(\omega) = H\left(\frac{\omega}{2} \right) \phi\left(\frac{\omega}{2} \right). \tag{2.33}$$

$H(\omega)$ is the discrete Fourier transform (DFT) of h_k. $H(0) = \mathrm{I}$ if $\phi(0) \neq 0$. This implies that the frequency response of $H(\omega)$ at DC is unity (low-pass filter).

Similarly, the Fourier transform of the mother wavelet is

$\hat{\psi}(\omega) = \int_{-\infty}^{\infty} \psi(t) e^{-j\omega t} dt$ and $G(\omega) = \sum_k \frac{g_k}{\sqrt{2}} e^{-jk\omega}$ then

$$\hat{\psi}(\omega) = G\left(\frac{\omega}{2}\right)\phi\left(\frac{\omega}{2}\right) \tag{2.34}$$

where $\hat{\phi}(\omega)$ and $\hat{\psi}(\omega)$ are the dilation and wavelet equation in the frequency domain, respectively.

But if the equation

$$\hat{\phi}(\omega) = H\left(\frac{\omega}{2}\right)\phi\left(\frac{\omega}{2}\right)$$

is iterated then

$$\hat{\phi}(\omega) = H\left(\frac{\omega}{2}\right)\left[H\left(\frac{\omega}{4}\right)\hat{\phi}\left(\frac{\omega}{4}\right)\right]$$

after n iteration, this becomes

$$\hat{\phi}(\omega) = H\left(\frac{\omega}{2}\right)\hat{\phi}\left(\frac{\omega}{2}\right)\ldots H\left(\frac{\omega}{2^n}\right)\hat{\phi}\left(\frac{\omega}{2^n}\right)$$

This leads to

$$\hat{\phi}(\omega) = \prod_{n=1}^{\infty} H\left(\frac{\omega}{2^n}\right) \tag{2.35}$$

2.10. Analysis of signals

The idea of decomposition and reconstruction are often used in signal processing. The decomposition or analysis process involves dividing the signal into different components for processing. The reconstruction (synthesis) process reconstructs the decomposed signal. An important issue here is the ability to reconstruct the decomposed signal perfectly. For the purpose of discussion, the following equations are repeated here;

$$f^j(t) \in V^j, \text{ and } f^j(t) = \sum_k a_k^j \phi_k^j(t) \tag{2.36}$$

$$g^j(t) \in W^j, \text{ and } g^j(t) = \sum_k d_k^j \psi_k^j(t) \tag{2.37}$$

Following the MRA requirement i.e., $V^{j+1} = V^j + W^j$ [Liu and Chang, 1998, pp.15], a function

$$f^{j+1}(t) \in V^{j+1}$$

can be written as;

$$\begin{aligned} f^{j+1}(t) &= f^{j}(t) + g^{j}(t) \\ &= \sum_k a_k^j \phi_k^j(t) + \sum_k d_k^j \psi_k^j(t) \end{aligned} \qquad (2.38)$$

$$\text{where } \phi_k^j = 2^{j/2}\phi(2^j t - k)$$
$$\psi_k^j = 2^{j/2}\psi(2^j t - k)$$

$2^{j/2}$ maintains the unity norm of the basis function [Burrus et al, 1998, pp.32].

Using the dilation equation

$$\phi(t) = \sum_n h(n) 2^{1/2} \phi(2t - n) \qquad (2.39)$$

and assuming a unique solution exists, then the scaling and translating of the time variable 't' gives

$$\begin{aligned} \phi(2^j t - k) &= \sum_n h(n) 2^{1/2} \phi(2(2^j t - k) - n) \\ &= \sum_n h(n) 2^{1/2} \phi(2^{j+1} t - 2k - n) \end{aligned} \qquad (2.40)$$

Using the variable $m = 2k + n$ [Burrus et al, 1998, pp.31] and substituting it into (2.40) gives

$$\phi(2^j t - k) = \sum_n h(m - 2k) 2^{1/2} \phi(2^{j+1} t - m) \qquad (2.41)$$

If V^j is denoted as

$$V^j = \underset{k}{Span}\{2^{j/2}\phi(2^j t - k)\} \qquad (2.42)$$

then

$$f(t) \in V^{j+1} \Rightarrow f(t) = \sum_n a_k^{j+1}(t) 2^{(j+1)/2} \phi(2^{j+1} t - k) \qquad (2.43)$$

is expressible at a scale of $j+1$ with scaling function only and no wavelets [Burrus et al, 1998, pp.32]. At a lower scale resolution, wavelets are necessary for the 'detail' not available at a scale of j. Therefore

$$f(t) = \sum_k a_k^j(t) 2^{j/2} \phi(2^j t - k) + \sum_k d_k^j(t) 2^{j/2} \psi(2^j t - k) \qquad (2.44)$$

If the $\phi_k^j(t)$ and $\psi_k^j(t)$ are orthonormal, the j level scaling coefficients are found by taking the inner product:

$$a_k^j(t) = \int_{-\infty}^{\infty} f(t) 2^{j/2} \phi_k^j(2^j t - k) dt = \langle f(t), \phi_k^j(t) \rangle \qquad (2.45)$$

and by using equation 2.41 gives

$$a_k^j(t) = \int_{-\infty}^{\infty} f(t) 2^{j/2} \sum_m h(m-2k) 2^{1/2} \phi(2^{j+1}t - m) dt$$

and interchanging the sum and integral, equation 2.45 can be written as

$$a_k^j(t) = \sum_m h(m-2k) \int_{-\infty}^{\infty} f(t) 2^{(j+2)/2} \phi_k^j(2^{j+1}t - m) dt \qquad (2.46)$$

$\int_{-\infty}^{\infty} f(t) 2^{(j+1)/2} \phi_k^j(2^{j+1}t - m) dt = a_k^{j+1}(m)$ giving

$$a_k^j(t) = \sum_m h(m-2k) a_k^{j+1}(m) \qquad (2.47)$$

The corresponding relationship for the wavelet coefficient is

$$d_k^j(t) = \sum_m h_1(m-2k) a_k^{j+1}(m) \qquad (2.48)$$

2.11. Digital Filter Interpretation

Equations 2.47 and 2.48 basically describe a digital filtering and down-sampling process. The two equations indicate that the expansion coefficient $a_k^{j+1}(m)$ is convolved with the time-reversed recursion coefficients $h_0(-n)$ and $h_1(-n)$ followed by a down-sampling process. The down-sampling process omits every other value of the signal to be sampled. The convolution and down-sampling process produce the approximation and detail coefficients at the next level $j-1$ [Burrus et al, 1998, pp.33, Villasenor et al, 1995].

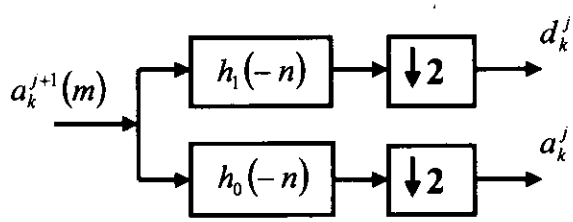


Figure 2.17 Two-band analysis bank

Figure 2.17 shows the implementation of equation 2.47 and 2.48 using a two-band analysis bank where $h_0(-n)$ and $h_1(-n)$ are finite impulse response (FIR) filters and the down arrows denote down-sampling. The FIR filter implemented by $h_0(-n)$ is a low-pass filter, and $h_1(-n)$ is a high-pass filter. The $a_k^{j+1}(m)$ coefficients are filtered by two FIR filters

with coefficients $h_0(-n)$ and $h_1(-n)$. The results are then down-sampled to give the next coarser scaling and wavelet coefficients.

The filter coefficients $h_0(-n)$ and $h_1(-n)$ can be replaced by h_0 and h_1 to denote the scaling function coefficients for the dilation equation 2.26. The frequency response of a digital filter is the discrete Fourier transform (DFT) of its impulse response coefficients $h(n)$. This is defined by

$$H(\omega)=\sum_{-\infty}^{\infty}h(n)e^{j\omega n}$$

Using this definition, the filter coefficients h_0 and h_1 can be re-written as $H_0(\omega)$ and $H_1(\omega)$ respectively. Using this notation Figure 2.17 can be re-drawn as

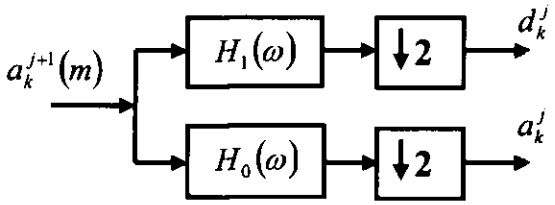


Figure 2.18 Two-band analysis bank re-drawn

The splitting and decimation process can be repeated to give a two-scale structure.

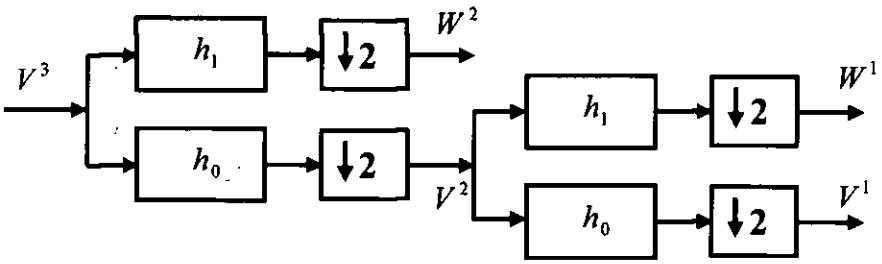


Figure 2.19 Three-stage two-band analysis tree

2.12. Synthesis of signals

The original signal can be reconstructed from a combination of the scaling function and wavelet coefficients at a coarse resolution. This can be derived by considering a signal in the $j+1$ scaling function space $f(t)\in V^{j+1}$ and the function can be written as;

$$f(t)=\sum_k a_k^{j+1}2^{(j+1)/2}\phi\left(2^{j+1}t-k\right) \tag{2.49}$$

or in terms of the next scale as

$$f(t)=\sum_k c_k^j 2^{j/2}\phi\left(2^j t-k\right)+\sum_k d_k^j 2^{j/2}\psi\left(2^j t-k\right) \tag{2.50}$$

Substituting equation 2.41 and the following

$$\psi(t)=\sum_n h_1(n)2^{1/2}\phi(2t-n),\quad n\in Z \tag{2.51}$$

into equation 2.50 gives

$$\begin{aligned} f(t)=&\sum_k a_k^j\sum_n h(n)\sqrt{2^{j+1}}\phi\left(2^{j+1}t-2k-n\right)+\ldots \\ &\sum_k d_k^j\sum_n h_1(n)\sqrt{2^{j+1}}\phi\left(2^{j+1}t-2k-n\right) \end{aligned} \tag{2.52}$$

Multiplying equation 2.49 and equation 2.52 by $2^{(j+1)/2}\phi\left(2^{j+1}t-k'\right)$ and integrating evaluates the coefficients as

$$a_k^{j+1}(k)=\sum a^j(m)h_0(k-2m)+\sum d^j(m)h_1(k-2m) \tag{2.53}$$

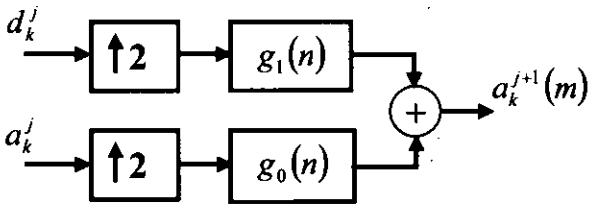


Figure 2.20 Two-band synthesis bank

2.13. Biorthogonal and Orthogonal Filter Bank

The combined analysis and synthesis structure with the associated scaling and wavelet functions is illustrated in the following diagram:

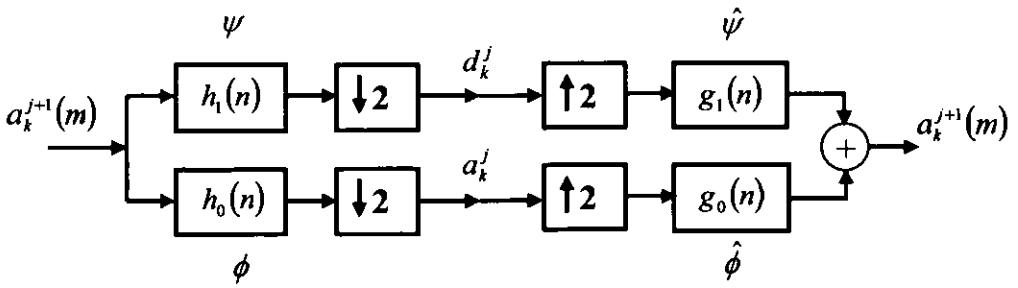


Figure 2.21 Filter bank structure

For a biorthogonal wavelet system the scaling function $\phi(t)$ and the dual $\hat{\phi}(t)$ are respectively defined by:

$$\phi(t)=\sqrt{2}\sum h_0(n)\phi(2t-n),\ \hat{\phi}(t)=\sqrt{2}\sum g_0(n)\hat{\phi}(2t-n)$$

and the associated wavelet function and its dual are defined by:

$$\psi(t)=\sqrt{2}\sum h_1(n)\phi(2t-n),\ \hat{\psi}(t)=\sqrt{2}\sum g_1(n)\hat{\phi}(2t-n).$$

For an orthogonal wavelet system:

$$\phi=\hat{\phi},\ \psi=\hat{\psi}$$

and

$$h_0(n)=g_0(n),\ h_1(n)=g_1(n)$$

2.14. Summary

The chapter began with a review of the Fourier series and showed how the Fourier transform is derived from the Fourier series. The drawback of the Fourier transform discussed in section 2.3 shows why the wavelet transform is preferred. The drawback is partially resolved by Gabor's STFT and is discussed in detail in section 2.4.

An alternative to the STFT, that is, the CWT is discussed in section 2.5 and 2.6. Two central ideas of wavelet theory, that is, scaling and translation are discussed in the same sections. Section 2.7 examines the DWT in detail and the connection between the DWT and filter banks is established in sections 2.11 and 2.12, which shows that the computation of the wavelet transforms can be accomplished by using filter banks.

The concept of MRA is explored in detail in section 2.8. The role of the scaling and wavelet function in MRA is discussed in detail. The scaling function is used to create a series of approximations of an image or function and the wavelet function is used to generate the details of an image, which contains the difference of information between adjacent approximations.

Lastly, the two main types of wavelets, orthogonal and biorthogonal wavelets are discussed in section 2.13. The discussion shows that a biorthogonal wavelet filter uses two wavelets instead of one wavelet as in an orthogonal wavelet filter. One wavelet is used for decomposition and the other for reconstruction. By using two wavelets, biorthogonal wavelets can be symmetric and have compact support [Hubbard, 1998, pp.243].

CHAPTER 3

Subband Coding, Wavelets, and Image Compression – A Review

3.I. Introduction

The first recorded mention of 'wavelet' dates back to 1909, in a thesis by Alfred Haar. However, the concept of wavelets in its present theoretical form was first proposed by Jean Morlet, a French geophysical engineer, in the late 1970's [Misiti, 1997, pp.1-29]. The theory of wavelet transforms has developed from the early 1980's. Since then wavelets have become very popular and are being studied increasingly with applications in digital signal processing and image processing.

The first orthogonal wavelets with compact support were constructed by Ingrid Daubechies [Daubechies, 1988]. These orthogonal wavelets were implemented using a filter bank algorithm, thereby establishing the link with filter banks and quadrature mirror filters (QMFs) which is also known as subband coding.

3.2. Subband Coding

Subband Coding (SBC) is a method of encoding audio signals efficiently. It takes advantage of the phenomenon of the human hearing system called 'masking'. The human ear is sensitive to a wide range of frequencies but when a lot of signal energy is present at one frequency, the ear cannot hear lower energy at nearby frequencies. When this occurs, the louder frequency is said to 'mask' the softer frequencies. SBC can save bandwidth by discarding information of the masked frequencies. The result is an approximation of the original signal. If the computation is done correctly, a human would not be able to detect the difference. The basic idea of SBC is to divide the frequency band of a signal into subbands and each subband is then coded with Pulse Code Modulation (PCM) or differential PCM (DPCM).

In the late 1970's Croisier et al [1976] and Crochiere et al [1976] developed the idea of subband coding using QMF to compress speech. Croisier et al employed a method based on interpolation-decimation-tree decomposition techniques to decompose the spectrum of a band-limited, Nyquist sampled, signal into a given number of adjacent channels. Each

channel is then moved and the sampling rate is reduced accordingly. The initial spectrum is reconstituted without frequency folding or distortion.

The interpolation-decimation-tree decomposition technique has a drawback, that is, aliasing, which occurs due to decimation. However, this drawback is overcome by the use of QMF. Croisier et al [1976] gives the following definition of QMF. Given a digital filter H_1, its quadrature mirror filter H_2 is defined as follows:

$$H_2(\omega) = H_1\left(\frac{\omega_s}{2} - \omega\right) \tag{3.1}$$

$$\phi_2(\omega) = \phi_1(\omega) \pm \frac{\pi}{2} \tag{3.2}$$

where $H_1(\omega) = H_1(\omega)e^{j\phi_1(\omega)}$ denotes the complex frequency response of the filter H_1, and $\omega_s = 2\pi f_s$ denotes the sampling rate.

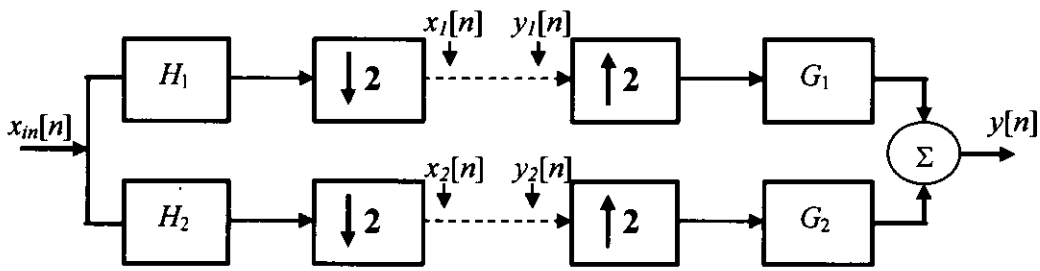


Figure 3.1 Two-channel QMF

With reference to Figure 3.1, the input signal $x_{in}[n]$ is filtered by H_1 and H_2. The filters H_1 and H_2 are known as an analysis filter bank (see section 2.11). Typically H_1 is a low-pass filter and H_2 is a high-pass filter. The two output signals $x_1[n]$ and $x_2[n]$ represent the low and high bands of $x_{in}[n]$. The resultant spectra occupy half the Nyquist bandwidth of the original signal and the sampling rate in each band is then down-sampled by a factor of 2. At the receiving end, $x_1[n]$ and $x_2[n]$ are up-sampled by a factor of 2 and passed through a two-band synthesis filter bank which consist of the filters G_1 and G_2 (see section 2.12). The outputs from these filters are then added to produce $y[n]$.

A similar implementation proposed by Crochiere et al [1976] uses a bank of non-overlapping band-pass filters. However, this approach suffered from aliasing effects caused by decimation and needed sophisticated band-pass filters.

3.2.1. Subband Coding of Images

As can be seen from the previous discussion, SBC originated in the context of speech processing, but it can be extended to image coding. Work related to image coding using subband coding can be traced back to the Kretzmer [1956] split-band quantisation method. Kretzmer's method was a high-frequency-low-frequency PCM-type approach. Another approach that is more directly related to subband coding of images is Burt and Adelson's [1983] Laplacian pyramid which produces, an approximate frequency decomposition. However, Burt and Adelson's approach did not explicitly use any subband coding concepts or QMF filters. Vetterli [1984] extended the QMF filter to a multidimensional case. It was Woods and O'Neil [1986] who extended subband coding to 2-D images (subband coding of images) using QMF.

With Woods and O'Neil's [1986] approach, the image is filtered with a 2-D QMF, and then the filter coefficients are sub-sampled. The resultant subbands, that is, LL, LH, HL, and HH are then formed.

Figure 3.2(a) shows how the input $x(m,n)$ is split into four subbands. Each subband is decimated by a factor of 2. After encoding, transmission, and decoding, the filtering process is carried out in reverse. See Figure 3.2(b).

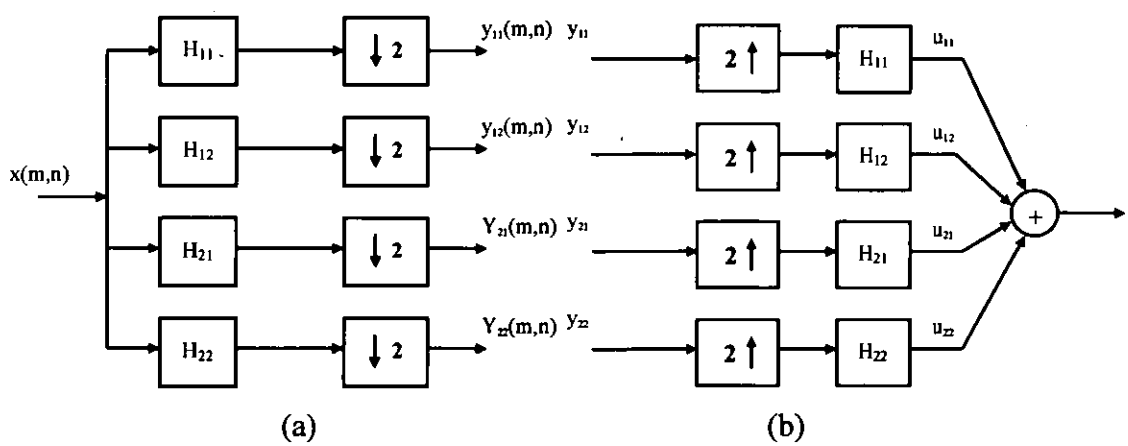


Figure 3.2 System diagram of 4-band splitting

Each subband is upsampled by a factor of 2 and bandpass filtered to eliminate aliased copies of the signal produced by the upsampling. The subbands are then summed to reconstruct the original signal.

The subbands are then coded with a DPCM scheme. The bits are optimally assigned among the subbands according to the DPCM error variances. An optimum mean square Laplacian quantizer is employed to quantize each subband.

The subband coded images are presented along with their signal-to-ratios (SNR's). The SNR performance of the subband coder is compared to the SNR performance of an adaptive DCT, vector quantization, and differential vector quantization for different bit rates. The results show that the adaptive subband coder has the best SNR performance.

3.3. Wavelet and Filter Bank

Mallat [1989a, 1989b] developed the idea of multiresolution analysis (MRA). MRA decomposes a discrete signal into frequency bands by a series of low-pass and high-pass filters to compute its DWT. MRA is very similar to the Croisier et al [1976] coding scheme. It could be said that historically, filters came first and then wavelets. It was Daubechies [1988] who discovered that QMFs can be used to generate wavelet bases. Since then, there have been a number of studies into wavelet and filter banks. This section will explore some of these works.

3.3.1. Mutliresolution

As mentioned earlier, Mallat initiated the idea of MRA which provides a natural framework for the understanding of wavelet bases. This section will explore Mallat's work on MRA.

Mallat's paper [1989a] on multiresolution analysis of signal decomposition describes a mathematical model for computing and interpreting multiresolution representation. In this study, Mallat shows the properties of an operator which approximates a signal for a given resolution. Mallat demonstrated that the difference in information between the

approximation of a signal at resolution 2^{j+1} and 2^j can be extracted by decomposing the signal on a wavelet orthonormal basis of $L^2(R^n)$.

Mallat's wavelet representation is calculated based on the decomposition of the original signal using a wavelet orthonormal basis. A wavelet orthonormal basis is a family of functions $\sqrt{2^j}\psi(2^j x - n)_{(j,n)\in z^2}$, that can be built by dilating and translating a unique function $\psi(x)$. This decomposition defines an orthonormal multiresolution representation called a wavelet representation. This representation can be computed with a pyramidal algorithm based on convolution using QMFs. The algorithm is illustrated by the following;

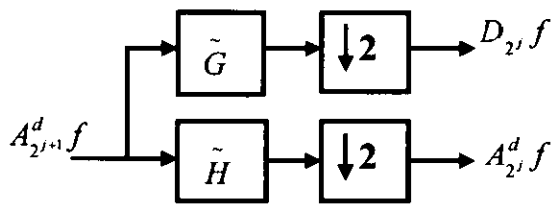


Figure 3.3 Decomposition algorithm

G and H are high-pass and low-pass filters respectively. G is the mirror filter of H and they are known as QMFs as described in section 3.2. $A^d_{2^j} f$ is known as the discrete approximation of $f(x)$ and $D_{2^j} f$ is the discrete detail signal at resolution 2^j. $D_{2^j} f$ contains the difference of information between $A^d_{2^{j+1}} f$ and $A^d_{2^j} f$.

The original signal can be reconstructed from the wavelet decomposition with a similar algorithm. The reconstruction algorithm is illustrated as follows:

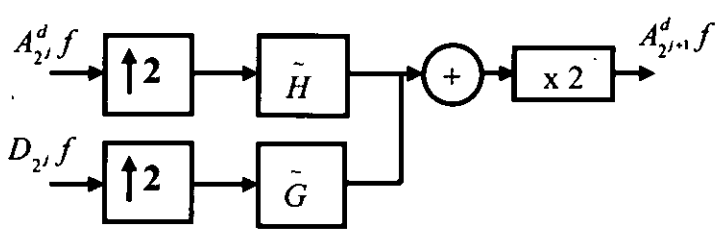


Figure 3.4 Reconstruction algorithm

This pyramidal algorithm based on convolutions with QMFs is very efficient. It can be observed that the above-mentioned decomposition and reconstruction algorithm is very similar to the subband coding of images proposed by Woods and O'Neil [1986]. However, it should be noted that both schemes were developed independently. The mathematical background of the decomposition and reconstruction algorithm has been discussed in sections 2.10, 2.11 and 2.12.

In Mallat's [1989a] study, decomposition and reconstruction algorithms in two dimensions is discussed. The two-dimensional decomposition algorithm is illustrated in Figure 3.5.

In Figure 3.5 $A^d_{2^{J+1}}f$ is known as the discrete approximation or reference signal of a signal $f(x)$ at the resolution 2^J. $D^1_{2^J}f$, $D^2_{2^J}f$, and $D^3_{2^J}f$ are detail signals at resolution 2^J. The image $A^d_{2^{J+1}}f$ is convolved with a one-dimensional filter and every other row is retained. Next, the columns of the resulting signal are convolved with another one-dimensional filter and every other column is retained. The filters used are the QMFs H and G described earlier.

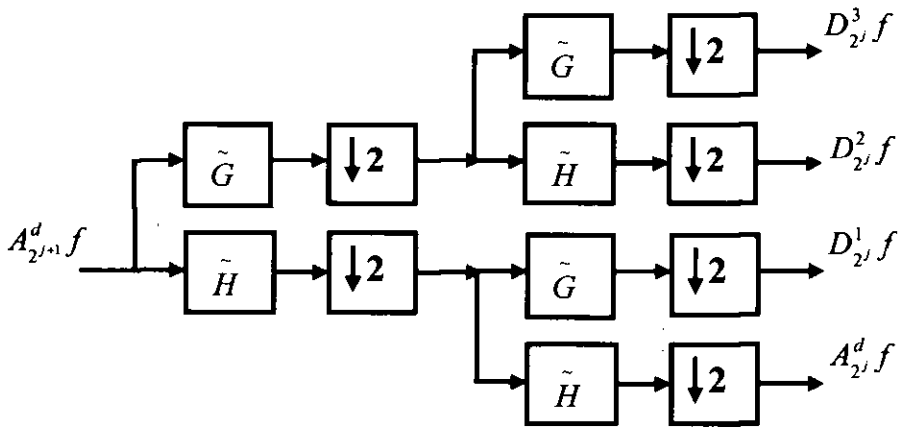


Figure 3.5 A two-dimensional decomposition algorithm

The following figure illustrates how the image $A^d_{2^{J+1}}f$ is decomposed into $A^d_{2^J}f$, $D^1_{2^J}f$, $D^2_{2^J}f$, and $D^3_{2^J}f$.

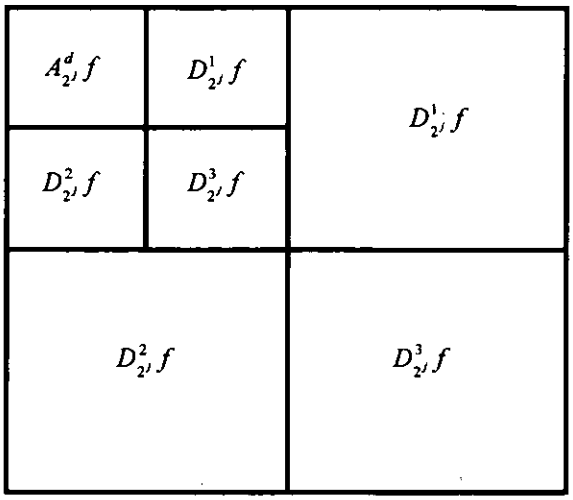


Figure 3.6 Decomposition of an image

The image $A^d_{2^j} f$ corresponds to the lowest frequencies, $D^1_{2^j} f$ gives the vertical high frequencies (horizontal edges), $D^2_{2^j} f$ the horizontal high frequencies (vertical edges), and $D^3_{2^j} f$ the high frequencies in both directions (diagonal).

The reconstruction algorithm is illustrated in the following figure;

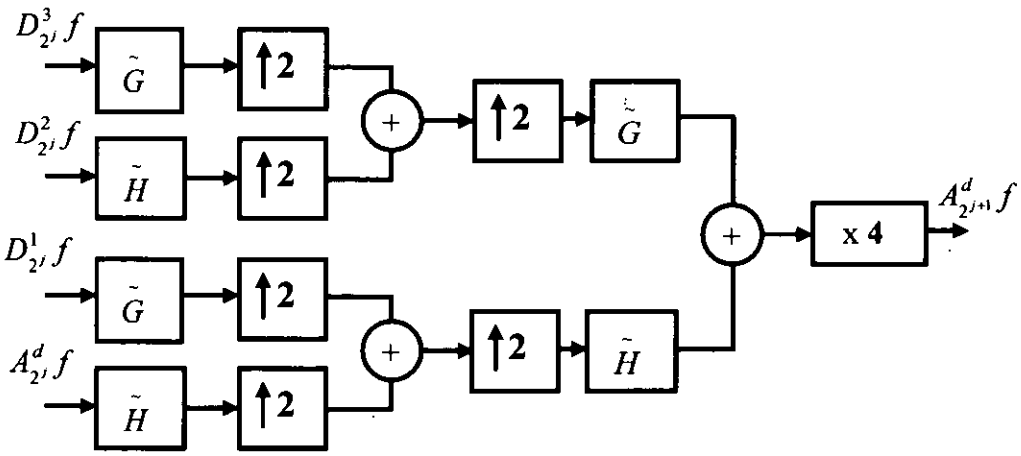


Figure 3.7 Reconstruction of an image

The image $A^d_{2^{j+1}} f$ is reconstructed from $A^d_{2^j} f$, $D^1_{2^j} f$, $D^2_{2^j} f$, and $D^3_{2^j} f$. Between each column of the images $A^d_{2^j} f$, $D^1_{2^j} f$, $D^2_{2^j} f$, and $D^3_{2^j} f$, a column of zeros is added and the rows are convolved with a one-dimensional filter. A row of zeros is added between each

row of the resulting image. Then the columns are convolved with another one-dimensional filter. The filters used are the QMFs *H* and *G* described earlier.

3.4. Still Image Compression Standards

Section 1.1 has shown that transmission of digital colour images using mobile devices such as mobile telephones can be slow and may be costly. Thus effective data compression techniques are essential for transmitting and storing digital colour images. The DCT-based JPEG standard has been the compression standard of choice for many years, but in recent years this has been overshadowed by the new JPEG2000 standard. JPEG2000 is a wavelet-based compression standard.

The following sections begin by exploring image compression methods, in particular JPEG. The JPEG2000 standard will also be explored in detail in section 3.9. This will help to better understand the reasons why wavelet transforms have become the transforms of choice for many image processing applications.

3.5. JPEG Standard

Much of the earlier work and research into image compression has produced image compression standards that give excellent compression performance in terms of compression ratio, peak signal to noise ratio (PSNR) and bits-per-pixel (bpp). The JPEG standard is an example, which is widely used to compress continuous tone or natural images.

JPEG is a lossy compression algorithm that exploits the human visual system ineffectiveness to perceive small changes in colour in an image. The algorithm first divides up an image into 8x8 pixels blocks and then converts the image from an RGB colour space to a luminance-chrominance colour space, such as YUV. The next stage in the JPEG algorithm is to apply a Discrete Cosine Transform (DCT) to the all 8x8 blocks. A two-dimensional DCT is used and it is defined by [Wallace, 1992]:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \quad (3.3)$$

where $C(u), C(v) = \frac{1}{\sqrt{2}}$ for $u, v = 0$;

$C(u), C(v) = 1$ for $u, v > 0$

With reference to equation (3.3), *f(x,y)* is the intensity or brightness of the pixel in row *x* and column *y*. *F(u,v)* is the DCT coefficient in row *u*, and column *v* of the DCT matrix. The output after the DCT process is a set of 64 basis-signal amplitudes or 'DCT coefficients', which contains integers ranging from -1024 to 1023. Much of the signal energy lies at low frequencies and these frequencies appear in the upper left corner (0, 0) of the DCT matrix. In principle, the DCT introduces no loss to the source image samples, it merely transforms the original 64-point signals to the frequency domain where they can be efficiently encoded.

JPEG suffers from blocking artefacts when compression is high and they are most visible at very low bit-rate. This artefact is primarily due to the negligence of correlations among adjacent blocks by a coarse quantization of DCT coefficients independently in these blocks [Yung et al, 1996]. Yung et al [1996] characterize and quantify the blocking artefact and then propose an iterative post-processing approach to remove the blocking artefact. A block classification scheme was developed to prevent over-smoothing in the edge and texture regions. Frequency domain filtering, which exploits the correlation of low frequency components among adjacent blocks was used. The attempt was broadly successful in improving the JPEG decoded image to achieve a better visual appearance, faster convergence rate and higher PSNR value but it has never been adopted or incorporated by any compression standard.

Vanhoucke [2001] applied a method called the Modified DCT (MDCT), which is used in perceptual audio coding, to overcome the blocking artefact. Using the Modified DCT for transform coding of images seems to be a very practical solution to the blocking artefact problem. Even at high bit rate, the smoothing effect can produce very high quality images. However, there are drawbacks to this method. The computational complexity can be

expected to increase by 50%, and the bit allocation strategy is made more challenging. Perhaps a faster algorithm to implement the MDCT might make it a more viable solution.

DCT-based compression schemes like JPEG utilize the fact that most natural images have light or smooth edges. Therefore, most blocks contain primarily low frequencies, and these blocks can be represented by a small number of coefficients without significant loss. However, with synthetic images the edges are sharp in contrast to a natural image. Sharp edges are associated with high frequencies. As a result, the DCT of the blocks where the edges pass has high-amplitude coefficients indicating high frequencies. These high frequencies cannot be removed without significant distortion.

The edges can be smoothed by adjusting the measures of block 'edginess' and image roughness, while restricting the DCT coefficient values to values that would have been quantized to those of the compressed image [Ahumada, 1995]. The basic steps of the algorithm are first to minimize the root mean square (RMS) error by adjusting the DCT coefficients and then calculating the block edge variance and finally, estimating what it should be. Next, the block edge variances are lowered to the estimated values from the previous step and then all the DCT coefficients are quantized to the coefficients of the compressed image.

All these methods seem to work well, but the computation overhead still remains high. This may be reduced by the use of a simple low-pass filter. However, this simple solution does have a drawback, that is, some information will be lost.

As mentioned earlier, the overall coding performance does not depend solely on the transform alone but on the two other components of an image coder, that is, the quantizer and entropy encoder. It would be logical to examine them to understand better their role in image compression.

3.6. Quantization

Quantization is a process where the amplitude of a signal is converted into a finite series of discrete amplitudes. In JPEG, the DCT coefficients are quantized in conjunction with a quantization table. The quantization table is specified as an input to the encoder. The quantization value is defined as:

$$F^{Q}(u,v) = IntegerRound\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{3.4}$$

where F^{Q} is the quantized value, $F(u,v)$ is the DCT coefficients, and $Q(u,v)$ is the value from a quantization table. The quantized values are rounded off and then normalized by the quantizer step size.

There are two types of quantization, scalar quantization (SQ) and vector quantization (VQ). In SQ, the input is processed individually to produce the output, while in VQ, the inputs are grouped together into vectors and processed to give the output.

3.6.1. Scalar Quantization

SQ is the simplest of all lossy compression schemes and it is used in the JPEG standard. Research on scalar quantization includes Ortega and Vetterli's [1997] novel technique for adaptive scalar quantization. The algorithm utilizes samples that were previously quantized, to estimate the distribution of the source. With this proposed scheme, the side information needed to adapt to changing source statistics does not need to be sent. Ortega and Vetterli [1997] proposed that the adaptive quantizer can be divided into two building blocks, model estimation and quantizer design. The purpose of the model estimation is to produce an estimate of the changing source probability density function. This estimate is used to redesign the quantizer. The results show that loss due to adaptivity can be minimized.

3.6.2. Vector Quantization

VQ is a lossy compression technique and can achieve higher compression ratios than SQ. A review of VQ methods for image coding can be found in Nasrabadi and King's paper [1988]. VQ has also been extensively investigated for image subbands [Cosman et al, 1995]. Cosman et al cited that subband/VQ coders are very competitive with subband/SQ coders that incorporate sophisticated lossless coding and both camps have reported improvement in performance. Cosman et al also cited that Shannon theory supports the distortion/rate superiority of VQ over SQ. However, the recent wavelet/subband coders like JPEG2000 does not support this view and instead has opted for a scalar quantizer in its implementation. Therefore, it remains to be seen whether VQ algorithms can achieve the advantages predicated by Shannon theory.

3.7. Entropy coding

Entropy coding is the last stage in the JPEG procedure. Additional compression is achieved by encoding the quantized DCT coefficients. The JPEG proposal specifies two entropy coding methods: Huffman Coding [Huffman, 1952] and arithmetic coding [Rissanen and Langdon, 1979]. The entropy coding step achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics.

A detailed description of the development of a method for the construction of a minimum-redundancy code is found in Huffman's paper. Huffman showed that a minimum-redundancy code or 'optimum code' can be constructed in such a way that the average number of coding digits per message is minimized. Huffman coding used in image compression is based on the frequency of occurrences of pixels in an image instead of symbols in a message. The pixels that occur frequently are encoded with a lower number of bits. A Huffman code table and the encoded pixels must be transmitted so that decoding can be carried out. A Huffman code table may have to be constructed for each image or for a set of images. Huffman tables may be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression [Wallace, 1992].

Unlike Huffman coding, the arithmetic coding method specified in the JPEG proposal requires no tables to be externally input, because it is able to adapt to the image statistics as it encodes the image [Wallace, 1992]. Arithmetic coding encodes the symbols using non-integer numbers of bits. It was the implementation of arithmetic coding by Moffat et al [1995], which incorporates several improvements over a widely used earlier version that has become a de facto standard. A paper by Howard and Vitter [1992] shows examples of how arithmetic coding achieves its performance. The paper points out that the main disadvantage of arithmetic coding is that it tends to be slow. Some of the other drawbacks of arithmetic coding are also highlighted. A fast coder, based on reduced-precision arithmetic coding was developed to overcome these problems is discussed. The next section explores research pertaining to wavelet-based compression.

3.8. Wavelet-based Compression

As mentioned above, JPEG suffers edge effects and blocking artefacts at high compression ratios. Different schemes have been proposed to reduce these artefacts and some of these schemes are discussed in the preceding section. In recent years, novel approaches like wavelet transforms have been used to resolve the problems encountered in JPEG-based compression. One such example is JPEG2000. JPEG2000 uses wavelet transforms instead of the DCT. The advantage of a wavelet transform is that, in contrast to JPEG, it does not divide the image into blocks, but analyses the image as a whole. Unlike the DCT used in the traditional JPEG, wavelet transforms allow a high compression ratio to be achieved and yet maintains the image quality. A recent study [Santa-Cruz et al, 2002] has shown not only does JPEG2000 provide superior rate-distortion performance but that there is significant improvement in functionality and it also provides lossy and lossless compression. Similar works are also found in two separate publications: Santa-Cruz and Ebrahimi, 2000a, Santa-Cruz and Ebrahimi, 2000b.

3.9. JPEG2000 Standard

JPEG2000 is a wavelet-based still image compression scheme and has several advantages over the original JPEG standard. Firstly, JPEG2000 compression efficiency at low bit rates is very much improved. In addition, new functionalities like multi-resolution representation, scalability and embedded bit stream architecture, lossy to lossless progression, and region-of-interest (ROI) coding are part of the standard [Adams, 2002].

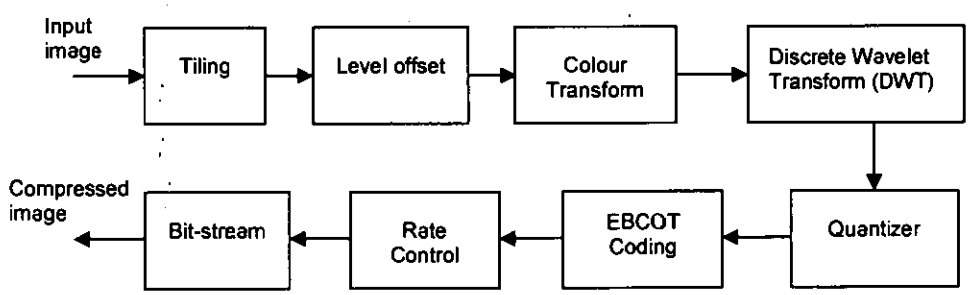


Figure 3.8 JPEG2000 encoding process

The tiling operation divides the input image into rectangular non-overlapping tiles or blocks as depicted in the following figure:

Figure 3.9 Tiling example

These tiles can be compressed independently. The tiling operation is optional and it is only used when the image to be encoded is larger than the encoder memory. Tiling can reduce memory requirements since each tile is reconstructed independently. In addition, specific parts of the image can be decoded, rather than the whole image thus reducing the memory requirements. This tiling process is similar to the JPEG compression scheme although the tile size is larger than the block size used in JPEG. The tiling process causes tiling artefacts at the tile boundaries at low-bit rate [Liang et al, 2003].

3.9.1. Level Offset

JPEG2000 requires that the input sample data should have a nominal dynamic range centred about zero. This requirement is the result of high-pass filtering used by JPEG2000. To ensure that input sample data should have a nominal dynamic range centred about zero, the level offset pre-processing stage is essential. Level offset is achieved by adding an offset of -2^{B-1} to the input sample so that the samples have a signed representation in the range of $-2^{B-1} \le x[n] < 2^{B-1}$ [Taubman et al, 2002, pp.418]. If the input sample is already centred about zero then no level offset is performed.

3.9.2. Colour Transform

Most colour images are commonly represented in RGB format, which is not suitable for image compression. In the RGB format, the image is composed of three independent grey-scale images, where each image corresponds to red, green and blue components. A more suitable colour format or colour space for image compression is the luminance/chrominance colour space. One such colour space is the YC_rC_b space, which is

statistically less dependent than the R, G and B colour components [Limb and Rubinstein, 1974]. JPEG2000 uses an irreversible colour transform (ICT) to transform a RGB image into a YC_rC_b image. ICT is defined as:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3.5)$$

3.9.3. Discrete Wavelet Transform

JPEG2000 uses a one-dimensional wavelet transform (1D DWT) to decompose each image tile into high and low sub-images. A two-dimensional decomposition is achieved by applying the 1D DWT along the horizontal and vertical axes of the image tile. This results in four subband images; low subband image (LL), high subband image (HL), low subband image (LH), and high subband image (HH). See Figure 3.10. This process can then be repeated with the LL image several times. The process is known as a dyadic decomposition.

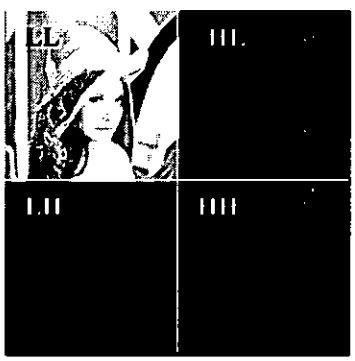


Figure3.10 Level decomposition

The DWT can be irreversible or reversible. The default irreversible wavelet transform is implemented by means of a Daubechies 9/7 tap filter [Christopoulous et al, 2000a; Antonini et al, 1992] and the default reversible transform is implemented by means of a LeGall 5/3 tap filter [LeGall and Tabatabai, 1988].

The JPEG2000 standard supports a convolution-based and lifting-based filtering mode. A series of dot products between the high-pass and low-pass filter coefficients and an

extended one-dimensional signal is performed in the convolution-based mode. In the lifting-based mode, the odd coefficient values are updated with a weighted sum of even coefficient values, and the even coefficient updated with a weighted sum of odd coefficient values.

3.9.4. Quantization

After the DWT, the wavelet coefficients are quantized. The quantization process reduces the precision of the coefficients. In Part 1 of the JPEG2000 standard, a scalar quantizer is used and it is defined as:

$$q_b[n] = sign(y_b[n]) \left\lfloor \frac{|y_b[n]|}{\Delta_b} \right\rfloor \tag{3.6}$$

where Δ_b is the quantization step size, and $y_b[n]$ is the transform coefficient for subband b [Taubman et al, 2002, pp.437]. The quantization process is lossy, unless the coefficients are integers as produced by the LeGall 5/3 tap filter. In the case where the coefficients are integers the quantization step size is set to 1, which means that no quantization is performed and the coefficients remain unchanged.

3.9.5. Coding

JPEG2000 uses a coding technique based on Embedded Block Coding with Optimal Truncation (EBCOT). Prior to coding, each subband of each tile is further partitioned into precincts and code-blocks. These code-blocks are essential entities used in coding. Each code block is compressed independently, a bit-plane at a time. For each bit-plane in a code-block, a special code-block scan pattern is used for each of the three passes, that is, the *significance propagation pass*, the *magnitude refinement pass* and the *clean-up pass* [Taubman et al, 2002, pp.363-367]. In the significance propagation pass, a bit is coded if its location is not significant, but at least one of its eight-connected neighbours is significant. In the magnitude refinement pass, all the bits from locations that became significant in a previous bit plane are coded. Lastly, in the clean-up pass, any bits not coded in the first two passes are taken care of.

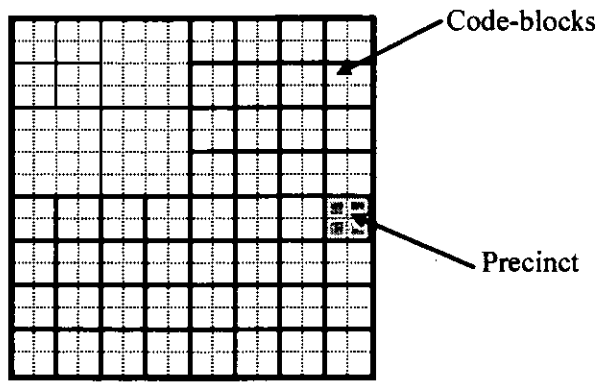


Figure 3.11 Tile partition into subbands, precincts and code-blocks

3.10. Region of Interest Coding

In region of interest (ROI) coding the background (BG) is coded with lower fidelity than the regions of interest (ROIs). In the wavelet domain ROI coding can be accomplished by identifying wavelet coefficients that affect the ROIs and these coefficients are then coded with a higher quality [Cruz et al, 1999].

The ROI coding scheme in Part 1 of the JPEG2000 standard is based on the MAXSHIFT method [Christopoulos et al, 2000b]. The MAXSHIFT method is an extension of Atsumi and Farvardin's [1998] work on a ROI scaling-based coding method. In the ROI scaling-based coding method the coefficients are scaled (shifted) so that they are placed in the higher bit-plane. Then, during the coding process, the most significant ROI bit-planes are placed in the bit-stream before any BG bit-planes of the image (see Figure 3.12). As a result, during the decoding stage the ROI bit-plane will be decoded first before the rest of the image [Christopoulous et al, 2000a].

In the MAXSHIFT method, the encoder scans the quantized coefficients and chooses a scaling value σ such that the minimum coefficient belonging in the ROI is larger than the maximum coefficient of the BG. During the decoding process every coefficient smaller than σ is scaled up.

To accomplish ROI coding an ROI mask is computed. The ROI mask is a map of the ROI in the image domain. It consists of non-zero value inside the ROI and 0 outside. The mask will thus indicate which coefficients are needed for the reconstruction of the ROI.

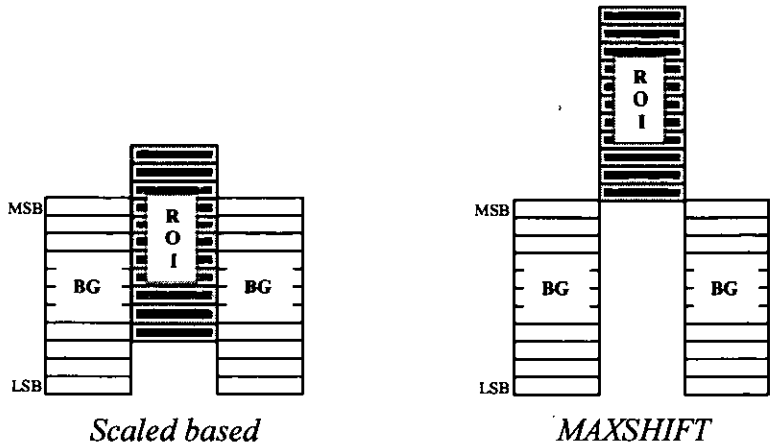


Figure 3.12 Comparing scaling-based method with MAXSHIFT method [source: Christopoulous et al, 2000a]

3.11. Summary

The chapter has shown that wavelets have proven to be a popular and powerful tool for signal analysis and synthesis. Wavelets have also found a wide range of applications especially in the area of image coding. The use of wavelet transforms in image coding was first discussed by Mallat [1989a] and Daubechies [1988]. Wavelet transforms can be mathematically related to subband coding [Vetterli, 1984, Woods and O'Neil, 1986]. Section 3.2 and section 3.3 examined work done in subband coding, wavelets and filter banks. The similarities between Woods and O'Neil's subband coding [1986] and Mallat's multiresolution [1989a] is remarkably striking. As already mentioned in section 3.3.1, both schemes were developed independently. In actual fact, it was Vetterli [1984] who first used QMF filter in a multidimensional case. Vetterli cited that separable filters are necessary and sufficient for the most natural four-band extension of the standard two-band QMF filters introduced by Croisier et al [1976] and Crochiere et al [1976]. However, Woods and O'Neil [1986] argued that separable filters are not necessary. It should also be noted that Vetterli's work contains no empirical data on image coding.

This chapter also examined the JPEG and JPEG2000 standards. This is necessary because it will help to better understand the short comings of JPEG and why wavelet-based methods are currently so popular in image compression. There are works that try to overcome the short comings of JPEG. Some of these works are discussed in Section 3.5. Work on entropy encoding methods like Huffman and arithmetic coding have also been explored.

CHAPTER 4

Wavelet-based Colour Image Compression Schemes

4.1. Introduction

There has been a variety of wavelet-based image compression schemes proposed over the last decade. This chapter will explore these significant schemes, which range from the simple to the more advanced schemes like Shapiro's [1993] embedded zero-tree wavelet scheme (EZW) and Said and Pearlman's [1996] set partitioning in hierarchical trees (SPIHT) coder. The more advanced schemes like the EZW and SPIHT use novel quantization and encoding techniques to improve the PSNR performance.

4.2. Wavelet-based Compression Schemes

Prior to JPEG2000, which uses a wavelet transform researchers have used wavelets to compress images. One such research is a JPEG-like image coder by Queiroz et al [1997].

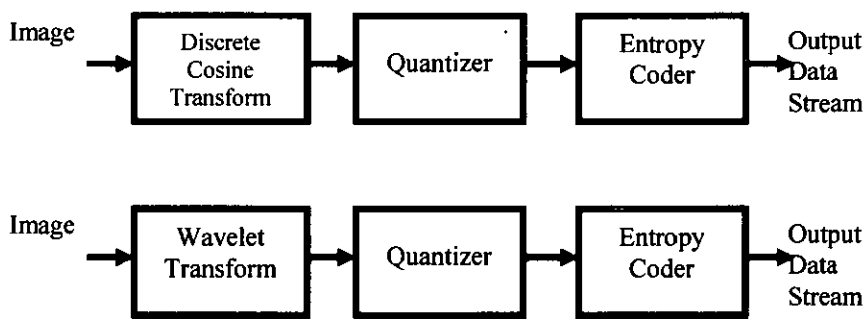


Figure 4.1 Queiroz et al strategy

The above is an illustration of Queiroz et al's [1997] strategy. The discrete DCT module is replaced with a DWT module. The rest of the JPEG components, that is the quantization and entropy encoder, remain the same. The DWT is realised as a two-channel filter bank.

The proposed coder by Queiroz et al [1997] adopted a procedure that groups the coefficients according to the subbands into blocks as shown in Figure 4.2. It is usual in a DWT that the coefficients are grouped and displayed by subbands as opposed to the 8x8 blocks of coefficients used in a DCT. In this study for n-levels, blocks of 2^n x 2^n samples

are constructed. The subbands are then scanned from low-frequency to high-frequency. The resulting block is scanned into a vector.

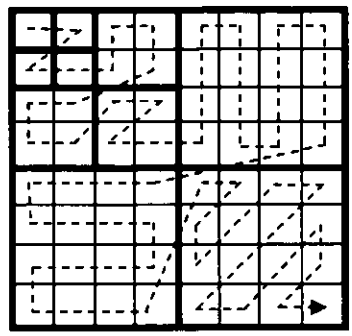


Figure 4.2 Scanning subbands [source: Queiroz et al, 1997]

This study showed that a DWT-JPEG scheme using existing JPEG components, except the transform, is possible. The tests have shown that the performance of the DWT-JPEG coder is close to that of sophisticated wavelet coders. The performance can be improved by improving the performance of the quantizer and Huffman encoder.

In a similar study, a 'new extended' JPEG coder [Huh and Hwang, 1997] was investigated to improve the performance of the JPEG extensions, which has been established in ISO/IEC 10918-3 (ITU_T Rec. T.84) [ISO, 1992]. The basic concept is to develop an efficient image coder combining the DWT features, while using variable quantization and entropy encoding and decoding procedures. In addition to all these, a new variable quantization matrix based on the human visual system (HVS) is generated. Like Queiroz et al [1997], Huh and Hwang [1997] replaced the DCT module with a block-wise discrete wavelet transform using filter banks. The block diagram for this proposed system is shown in Figure 4.3.

The reported simulated results demonstrate the improved performance of the DWT-based JPEG extension coder over the conventional JPEG coder. The performance of the DWT-based JPEG extension coder gives better objective and subjective quality at high compression.

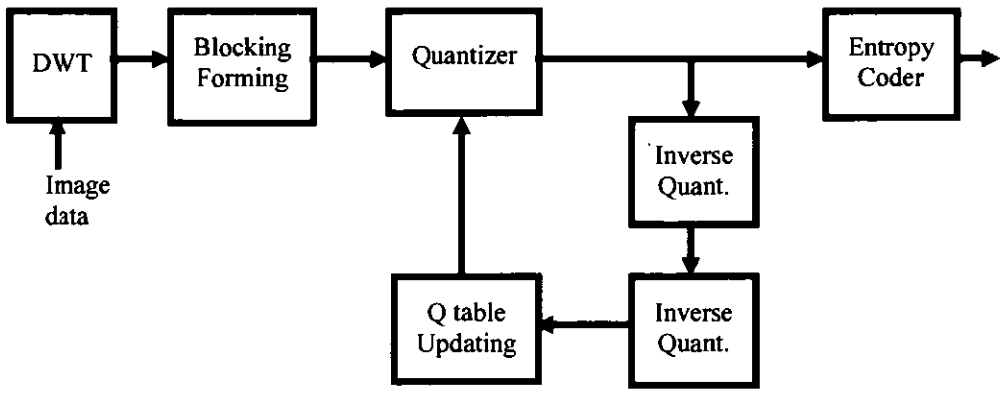


Figure 4.3 Huh and Hwang [1997] proposed coder block diagram

The results from the two studies show that the coding gains of the coders can be improved by replacing the DCT module with a wavelet transform. However, the coding gain does not depend solely on the transform used but other components of the image coder like quantization and entropy coding, which are equally important.

4.3. Spatial Oriented Tree

The EZW algorithm uses the 2-D discrete wavelet transform to decompose images into subband images for efficient quantization and encoding at different scales. The highest energy coefficients can be found in the lowest frequency subband images. Figure 4.4 shows a discrete wavelet transform two level decomposition.

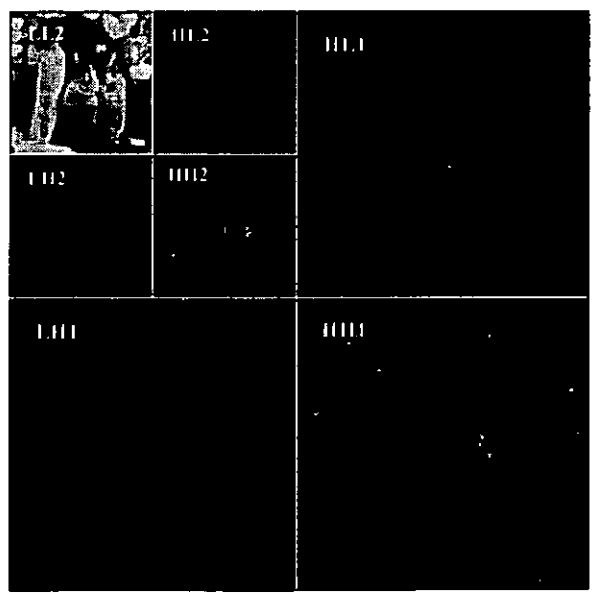


Figure 4.4 Subband images

The LL subband (approximation) contains most of the significant information and the other subband images (details) primarily consist of information of the edges.

An examination of subband HL2 of Figure 4.4, reveals that the subband consists mainly of a large region of black, which indicates zero coefficients. Examining HL1 shows a corresponding region of zero coefficients. The LH and HH subbands also have similar spatial regions. Figure 4.5 shows another way of looking at the subband decomposition:

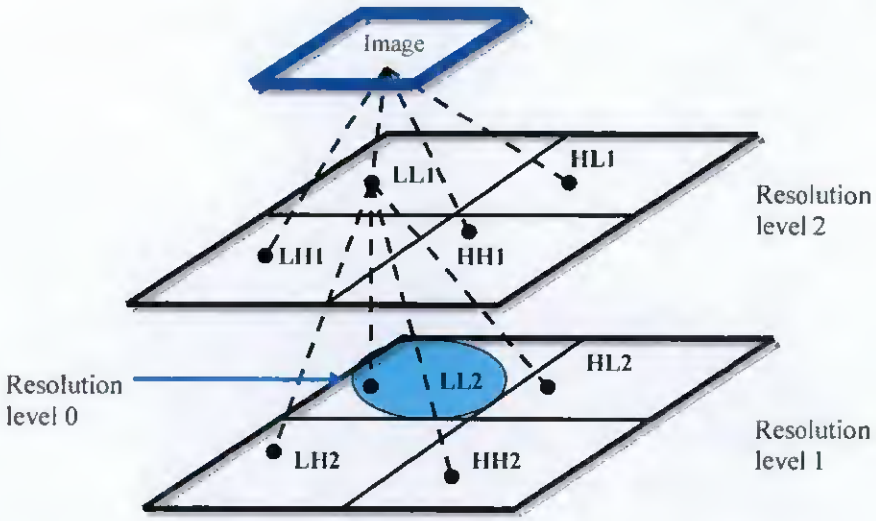


Figure 4.5 Subband decomposition [Adapted from Taubman et al, 2002]

The wavelet coefficients of various subbands having the same spatial location can be grouped into one structure known as a spatial orientation tree (SOT) [Rao and Bopardikar, 1998, pp.158-159]. The underpinning idea of SOT is based on the observation that most of the energy is found in the lowest subbands of a hierarchical decomposition. If the magnitude of the wavelet coefficients are insignificant with respect to a given threshold level, then it is likely that wavelet coefficients of the same spatial location in different subbands will also be insignificant. This relationship can be exploited by coding the root wavelet coefficient to which the entire set of coefficients is related and the set of coefficients is know as a zero-tree.

The genealogy of subband coefficients can be visualized in Figure 4.6. With reference to Figure 4.6, each coefficient in subband HH2 is considered to be the parent of four children

in subband HH1. This parent-child relationship applies to the LH and HL orientations. In other words, in a hierarchical subband system every coefficient at a given resolution can be related to a set of coefficients at the next finer resolution of similar orientation. The coefficients in the lowest frequency subband, that is LL2, each have only three children, comprised of the single coefficient at the same spatial location in HL2, LH2, and HH2.

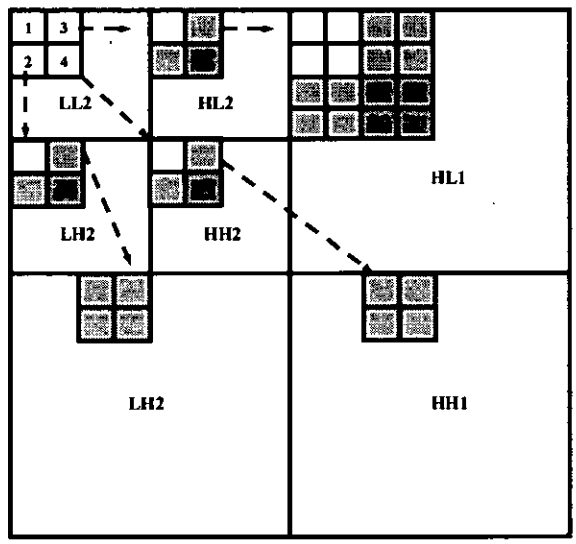


Figure 4.6 Zero-tree structure [source: Shapiro, 1993]

The lowest frequency subband is the top left, and the highest frequency subband is at the bottom right.

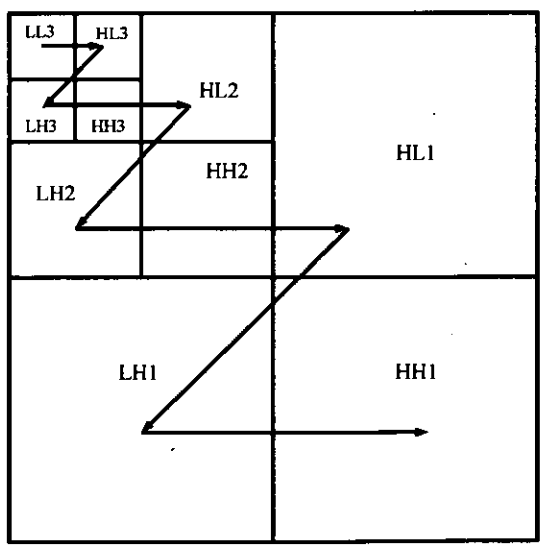


Figure 4.7 Scanning order of subbands [source: Shapiro, 1993]

The next step is scanning the coefficients. The order of scanning should be done in such a way that no child node is scanned before its parent. Figure 4.7 illustrate the order of scanning. All the positions in a given subband are scanned before moving on to the next subband.

4.4. EZW Algorithm

The EZW algorithm [Shapiro, 1993] begins by setting a threshold level T to determine whether or not a coefficient is significant. A wavelet coefficient c is said to be insignificant with respect to a given threshold T if $|c|<T$, otherwise the coefficient is said to be significant. For a threshold value T, a coefficient is an element of a zero-tree if the coefficient of concern and all of its descendents are insignificant with respect to the threshold T. If a bit-plane coding is adopted then the initial threshold T will be:

$$T = 2^{\lfloor \log_2 (Max(|c(x,y)|)) \rfloor} \tag{4.1}$$

where $Max(.)$ is maximum coefficient value and $c(x,y)$ is the coefficient [Valens, 2004]. $\lfloor\ \rfloor$ represents the largest integer less than $|c(x,y)|$ [Rao and Bopardikar, 1998, pp.160].

The image is coded using two passes, which are known as the dominant and subordinate passes. In the dominant pass, the image is scanned and a symbol is assigned to every wavelet coefficient. The symbols are: zero-tree root (ZRT), isolated zero (IZ), positive significant (POS) and negative significant (NEG). If the wavelet coefficient is greater than the threshold level T then a POS is assigned to the wavelet coefficient. A NEG is assigned to the wavelet coefficient if the wavelet coefficient is smaller than minus the threshold level T. If the coefficient is the root of a zero-tree then a ZRT is assigned. An IZ is assigned to a wavelet coefficient if the coefficient is smaller than the threshold. Each time a wavelet coefficient is encoded as significant (positive or negative), its magnitude is added to a subordinate list and the coefficient concerned is set to zero.

The dominant pass is followed by the subordinate pass in which all coefficients on the subordinate list are scanned. The width of the effective quantizer step size, which defines an uncertainty interval for the true magnitude of the coefficient, is cut in half. A coefficient in the upper half of the range gets assigned with a '1' symbol (for the upper part

of the range), while a coefficient in the lower half gets assigned with a '0' symbol. During the subordinate pass a string of symbols from this binary alphabet is generated, which is then entropy encoded.

The whole process continues to alternate between dominant and subordinate passes where the threshold is halved prior each dominant pass and can be summarised by the following pseudo-code:

```
threshold = InitialThreshold;
do
  {
     dominant_pass(image);
     subordinate_pass(image);
     threshold = threshold/2;
  } while (threshold > MinimumThreshold);
```

Shapiro reported that the EZW algorithm produces compression results that are competitive with a known wavelet-based compression algorithm developed by DeVore et al [1992]. The EZW was also compared to JPEG using the 'Barbara' test image as a reference. The results show that the EZW coder gives a significantly higher PSNR. Artefacts were reported for low bit rates. However, Shapiro cited that these artefacts are not as objectionable as the blocking effects produced by block transform coding schemes.

4.5. SPIHT

The Said and Pearlman's [1996] SPIHT (pronounced as 'spite') algorithm uses the set partitioning principles to improve the PSNR performance. The SPIHT algorithm has several features in common with the EZW algorithm.

The SOT for SPIHT is the same as those for EZW except for the LL2 subband. The SPIHT SOT is based on the assumption that the dimension of the LL subband is an integer multiple of two. Hence the coefficients in the LL subband are divided into groups of two-by-two. One fourth of the coefficients in LL have no children, while the remaining coefficients each have four children. Figure 4.8 shows a situation for a group of four coefficients in the LL2 subband. The coefficient marked with an asterisk has no children

(see Figure 4.8). All the other coefficients, except for those in the highest frequency bands have four descendents.

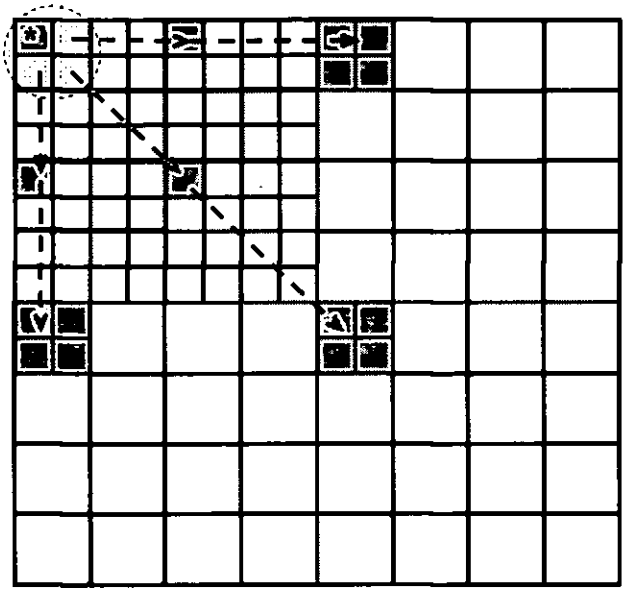


Figure 4.8 SPIHT SOT

In the SPIHT algorithm, the wavelet coefficients are grouped into three ordered lists, the list of significant pixels (LSP), the list of insignificant pixels (LIP) and the list of insignificant sets (LIS). LSP contains the coordinates of all pixels that are significant with respect to a threshold. LIS contains the coordinates of the roots of insignificant sets of pixels and lastly LIP contains a list of coordinates of all pixels that are insignificant, but do not reside within one of the two types of mentioned.

In the first stage of the coding algorithm, the output $n = \lfloor \log_2 \left(\max_{(i,j)} |c_{i,j}| \right) \rfloor$ is sent to the decoder. The value n is for testing significance of pixels and constructing the significance map. Next the LSP is set as an empty list and this is followed by adding a set of coordinates of all spatial orientation tree roots to the LIP. The set of coordinates with descendants are added to the LIS. The sorting pass and the refinement pass are then initiated.

During the sorting pass each pixel of the LIP is tested for significance with respect to n. If the pixel is significant then a 1 is transmitted and its coordinates are moved to LSP. A sign bit representing that pixel is also transmitted. The algorithm then tests each entry in the LIS

for the existence of significant descendents. If no significant descendents are found then a 0 is transmitted. However, if the entry has one significant descendent then a 1 is transmitted and each entry of the immediate descendents is tested for significance. If a descendent is found to be significant then a 1 and a sign bit are transmitted and the pixel coordinates tagged onto the LSP. If a descendent is found to be insignificant then a 0 is transmitted and the pixel coordinates are tagged onto the LIS. A pixel is moved to the end of the LIS as an entry of type B, if it has more descendents. The descendents of an entry in the LIS of type B are tested for significance. If at least one of them is significant, then this entry is removed from the list and its immediate descendents are tagged to the end of the list as entries of type A.

After a pass through the LIS is completed, a refinement pass through the LSP is initiated. The n^{th} most significant bit of each entry of the LSP is transmitted. The entry added in the current sorting pass is excluded. Next the quantization step is updated, where n is decremented by 1 and the sorting process is repeated. The output of this process is then handed over to an arithmetic encoder.

Said and Pearlman [1996] reported that the results obtained from the SPIHT algorithm surpasses those from Shapiro's original EZW.

4.6. Compressing Colour Images

The compression schemes described in the earlier sections are intended for grey-scale images. Wavelet-based colour image compression schemes will be explored in the later sections but works on coding of colour images prior to wavelets will be explored in this section. Work related to colour image coding can be traced to Limb and Rubinstein's [1974] method called plateau coding for coding chromatic component of colour television signals. Since most changes in chrominance are accompanied by luminance changes Limb's method exploits this effect by only transmitting new chrominance samples when there is a significant change in the luminance signal.

Referring to the following diagrams, the luminance signal is coded digitally using pulse-code modulation (PCM) or differential pulse-code modulation (DPCM) for transmission over the channel. The decoded luminance signal is used to select changes in the amplitude

of the signal. The averaging circuits average the signals between the detected significant changes in the luminance signal. The average signal is coded and read into the buffer and multiplexer under command of the change detector. At the decoder end the same detection process is repeated on the decoded luminance signal. The buffer and demultiplexer then output the coded averaged chrominance signal enabling the chrominance signals to be reconstructed.

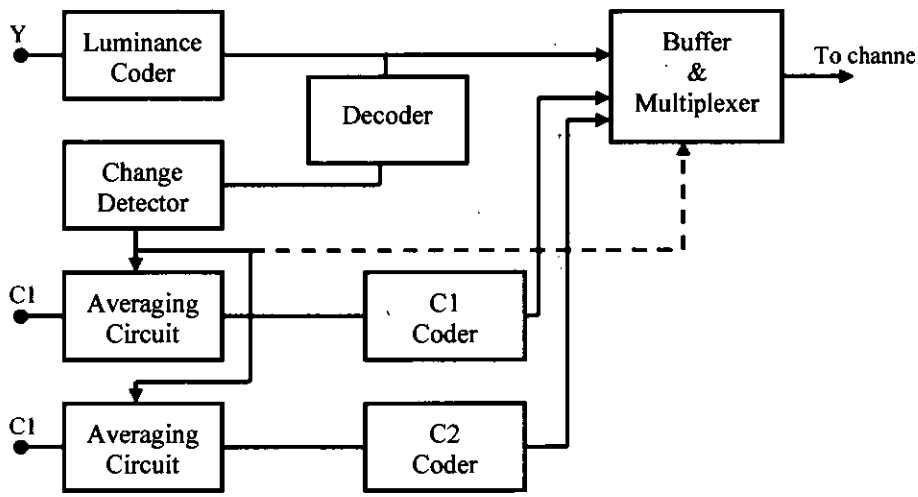


Figure 4.9 Block diagram of a plateau coder [source: Limb and Rubinstein, 1974]

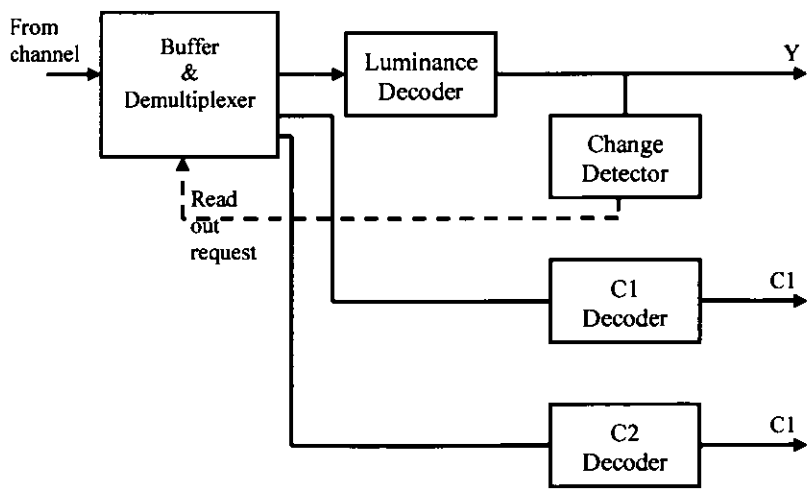


Figure 4.10 Block diagram of a plateau decoder [source: Limb and Rubinstein, 1974]

Simulations have shown that plateau coding can give pictures of high quality for chrominance bit-rates in the range 0.25 to 0.5 bits per luminance sample.

A follow up work by Netravali and Rubinstein [1979] also exploits the effect that the spatial changes in the chrominance coincide with spatial changes in the luminance. Netravali and Rubinstein proposed two techniques for digital coding of the chrominance components of a colour television signal. Both techniques are based on an observation make by Limb and Rubinstein that the spatial changes in the chrominance coincide with spatial changes in the luminance. Netravali and Rubinstein [1979] use this to predict the chrominance samples efficiently using the previously transmitted chrominance and luminance samples, and the present luminance sample.

Based on the results of computer simulations, Netravali and Rubinstein present two coding schemes. In the first scheme the adaptive prediction of the chrominance components is based on the luminance and the chrominance components are coded by a DPCM coder. The second scheme uses the chrominance signal to adaptively extrapolate its past using the luminance signal for adaptation. Only those chrominance samples where the extrapolation error is more than a threshold are transmitted to the receiver. Computer simulations on video telephone pictures indicate that, for the predictive coding, the entropy of the coded chrominance signals can be reduced by about 15 to 20 percent by adaptation.

4.7. Subband Coding of Colour Images

Section 3.2.1 discussed the work of subband coding of monochrome images using QMF by Woods and O'Neil [1986]. The application of subband coding to colour images was considered by Gharavi and Tabatabai [1988]. In this work, the R, G, and B components were transformed into luminance Y and chrominance signals I and Q. The luminance signal was treated as a monochrome image and the chrominance signals I and Q components, upper (high) band signals, were discarded. The simulated results demonstrated that subband coding can be an effective method to compress colour images.

4.8. Wavelet-based Colour Image Compression

This section will explore wavelet-based compression methods applied to colour images. One such method is a modified EZW for colour images known as Colour Embedded Zerotree Wavelet (CEZW) proposed by Shen [Shen, 1997, pp.57-78; Shen and Delp,

1997]. The SOTs discussed in sections 4.3, 4.4 and 4.5 are primarily designed for compressing monochrome images. The interdependency between the colour components were not exploited by the SOTs. Shen's approach used a modified SOT to exploit the correlation between the colour components of the YUV colour space.

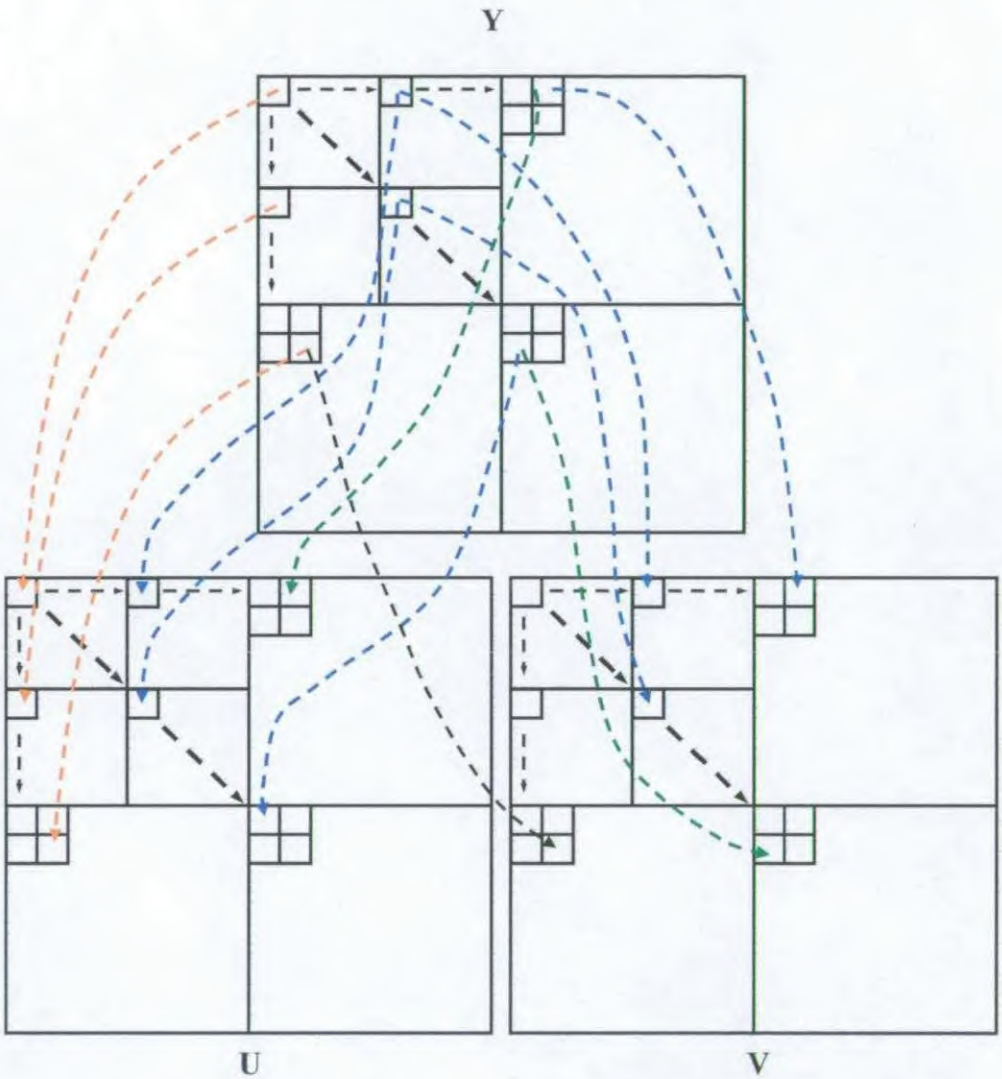


Figure 4.11 Parent-descendent relationships in the CEZW algorithm. [Adapted from Shen, 1997]

The following is a summary of the CEZW taken from Shen's thesis;

1. Wavelet transform is performed on each of the three colour components separately.

   Let T be the largest magnitude in wavelet transform coefficients c.

2. While bit budget is not exhausted

(a) $T = \frac{1}{2T}$

(b) Dominant pass:

i. The Y component is scanned and for each Y node its children in the Y component as well as those in the U and V components are compared with T. Symbols of Positive significant (POS), Negative significant (NEG), Zerotree (ZT) and Isolated Zero (IZ) are assigned and entropy coded.

ii. The U and V components are alternately scanned. The coefficients and their children nodes are compared with T. Those coefficients that have been coded as part of a zerotree in step (i) are not examined. Symbols of POS, NEG, ZT and IZ are assigned and entropy coded.

(c) Subordinate pass:

The coefficients that have been coded as significant in previous passes (excluding the dominant pass just preceding this subordinate pass) are examined. The quantization error is compared with T and symbols of Significant (SIG) and Insignificant (INS) are assigned and entropy coded.

The proposed CEZW algorithm was evaluated using Daubechies 9/7 tap filter bank. The entropy encoder used is an adaptive arithmetic coder. The PSNR ratios from the CEZW were compared with that from the SPIHT and JPEG. Table 4.1 shows the PSNR of the decoded images using CEZW, SPIHT and JPEG.

The results in Table 4.1 show that the PSNR from CEZW is 4 dB higher than that from SPIHT at all data rates. However, it should be noted that the PSNR values are biased in favour of CEZW using a luminance/chrominance colour space (YUV) instead of the RGB colour space. The PSNR is obtained from:

$$PSNR = 10\log_{10}\frac{255^5}{(MSE(Y)+MSE(U)+MSE(V))/3}$$

[Shen, 1997]. Therefore, it is difficult to ascertain the actual PSNR improvement over SPIHT and JPEG.

Table 4.1 PSNR of decoded image using CEZW, SPIHT and JPEG [Adapted from Shen, 1997]

Rate (bpp)		Images				
		girls	lenna	model	peppers	tiger
0.5	CEZW	36.1	36.9	37.4	41.9	31.9
	SPIHT	32.3	33.3	33.2	37.8	27.9
	JPEG	29.7	30.3	30.8	34.6	25.1
1.0	CEZW	39.3	40.1	40.5	44.8	35.1
	SPIHT	35.7	35.9	36.1	39.8	31.4
	JPEG	33.2	33.5	33.8	37.2	28.8
1.5	CEZW	42.3	42.7	43.0	45.9	38.3
	SPIHT	38.0	37.6	38.2	42.3	34.0
	JPEG	35.0	35.2	35.7	38.4	31.2

The performance of the SOT from the CEZW algorithm was evaluated by Saenz et al [1999] as part of an investigation into colour embedded image compression. The investigation was carried out by means of a set of core experiments that seek to evaluate the advantages of various colour transformations, SOTs and the use of monochrome embedded coding schemes such as EZW and SPIHT. The SOT from CEZW is used with various types of colour transformations. The colour transformations used were RGB to YUV and RGB to Karhunen-Loeve transform (KLT). The SOT from the CEZW algorithm was referred to as CZW. As in Shen [1997] a 9/7 biorthogonal wavelet filter and adaptive arithmetic coding was used in Saenz et al's evaluation. Each colour component of the test images were compressed independently using two new SOTs referred to as Naive-EZW (NEZW) and Naive-SPIHT (NSPIHT), respectively. Other techniques used were SP-CZW and CSPHIT. SP-CZW adapted the SOT from SPIHT and is used in conjunction with Shapiro's dominant and subordinate passes. CSPHIT is the SPIHT used to compress colour images. Some of the results from Saenz et al's evaluation are presented in Table 4.2.

Saenz et al concluded that the CZW and SP-CZW performance are similar and the best performance was obtained using CSPIHT. Saenz et al observed that the quality of the decompressed YUV and KLT images are better than that of the RGB images for all

techniques. In addition, Saenz et al also observed that the YUV images are visually better than the KLT images despite the fact that KLT perform better in terms of PSNR.

A closer examination of the results shows that in the YUV colour space at a bit rate of 1.5 bpp, the NSPIHT achieves the best PSNR value. The PSNR difference ranges from 0.39 to 0.9 dB. However, at a bit rate of 0.5 bpp, the SP-CZW achieves the best PSNR value and the PSNR difference ranges from 0.01 to 1.79 dB.

Table 4.2 PSNR results for the Barbara image [Adapted from Saenz et al, 1999]

Rate (bpp)	Image	Barbara (PSNR dB)					
		NEZW	NSPIHT	CZW	SP-CZW	JPEG	CSPIHT
0.5	RGB	23.63	24.59	24.12	24.12	24.51	-
	KLT	24.05	24.91	26.58	26.58	-	27.54
	YUV	24.68	26.01	26.46	26.47	-	-
1.0	RGB	25.36	26.66	26.92	26.92	26.43	-
	KLT	25.83	27.24	28.68	28.69	-	29.35
	YUV	27.18	28.23	28.50	28.50	-	-
1.5	RGB	27.05	27.99	27.32	27.42	27.23	-
	KLT	27.50	28.68	29.28	29.33	-	30.25
	YUV	28.59	29.49	29.10	29.10	-	-

4.9. Multiwavelets and Colour Image Compression

The advanced DWT-based compression schemes like Shapiro's EZW [1993] and Said/Pealman's SPIHT [1996] discussed in earlier sections utilizes scalar wavelets. In recent years balanced multiwavelets (BMW) have been used to compress colour images effectively. However, their performance is marginally short of the 9/7 tap biorthogonal scalar wavelets (SW). In a study carried out by Iyer and Bell [2001] the properties of the 9/7 tap biorthogonal SW ($B_{9/7}$) and four multiwavelets that are important to grey-scaled images were investigated. The four multiwavelets used are SA_4^u, SA_4^b, Bmw_8, and Bmw_{12}. SA_4^u is an unbalanced symmetric-antisymmetric multiwavelet and SA_4^b is a balanced

symmetric-antisymmetric multiwavelet. Bmw_8 and Bmw_{12} are both balanced multiwavelets. Partial results from Iyer and Bell's study are tabulated in Table 4.3 for three standard grey-scale images.

The results from Table 4.3 show that the $B_{9/7}$ wavelet achieves 0.09-1.3 dB higher PSNRs than SA_4^b. From the same table, the results also show that the unbalanced multiwavelet SA_4^b PSNR is about 0.26-1.11 dB worse than the B_{mw8} balanced multiwavelet. To appreciate these gains and losses, the images have to be magnified.

Table 4.3 PSNR results [Adapted from Iyer and Bell, 2001]

Image	CR	$B_{9/7}$	SA_4^u	SA_4^b	Bmw_8	Bmw_{12}
barbara	16:1	30.82	29.58	30.41	30.19	30.40
	32:1	27.04	26.27	26.95	26.71	26.93
goldhill	16:1	32.46	31.89	32.24	32.15	31.92
	32:1	30.13	29.34	29.88	29.74	29.54
lena	16:1	36.71	36.38	35.39	36.25	36.33
	32:1	33.59	31.97	33.21	33.08	33.18

The study of Iyer and Bell [2001] was extended to cover colour images by Rout and Bell [2002]. In the Rout and Bell study, the compression performance of three scalar wavelets and five balanced multiwavelets on seven colour images were investigated. Each RGB image is transformed into an YC_bC_r representation. A five level decomposition is used and the SPIHT is used to quantize the wavelet coefficients. No entropy encoding was used. Table 4.4 shows the PSNR results for three out of the seven images.

For each compression ratio, the maximum PSNR values are highlighted for both the SW and BMW. In terms of PSNR performance, the SWs perform better than the BMWs. The range of difference is 0.04 to 0.47 dB.

Table 4.4 PSNR results from Rout and Bell [2002]

	Scalar Wavelet				Multiwavelets				
Image	**CR**	**$B_{9/7}$**	**$B_{22/14}$**	**D_8**	**SA_4**	**Ort_4**	**Bmw_8**	**Bmw_{12}**	**Bmw_S**
lena	32:1	32.06	**32.19**	31.49	31.94	**31.97**	31.80	31.90	31.74
	48:1	30.81	**30.92**	29.76	30.42	**30.45**	30.24	30.38	30.14
lighthouse	32:1	**27.62**	**27.62**	26.80	**27.54**	**27.54**	27.46	27.52	27.41
	48:1	26.07	**26.18**	25.43	**26.05**	26.03	25.94	25.99	25.92
mandrill	32:1	22.06	**22.10**	21.93	**22.04**	22.03	21.93	22.00	21.97
	48:1	21.18	**21.21**	21.08	**21.17**	21.15	21.07	21.13	21.06

4.10. Summary

This chapter explored a variety of wavelet-based image compression schemes in particular advanced schemes like EZW and SPIHT. Some of the early wavelet-based works can be traced back to the works of Queiroz et al [1997] and Huh and Hwang [1997]. The two mentioned works basically replaced the DCT module with a discrete wavelet transform using filter banks. The reported results from Queiroz et al [1997] and Huh and Hwang [1997] show the DWT-based JPEG extension coder gives better objective and subjective quality at high compression.

Section 4.4 and 4.5 explored advanced wavelet-based compression schemes like EZW and SPIHT. The EZW is a very effective and computationally simple image compression technique. It is based on four key concepts, namely, (1) a discrete wavelet transform, (2) prediction of the absence of significant information across scale, (3) entropy-coded successive-approximation quantization and (4) lossless compression using adaptive arithmetic encoding. Like the EZW, the SPHIT is based on the zerotrees technique, which takes advantage of the structural similarities across the sub-bands when using wavelet decomposition of an image.

Wavelet-based colour image compression schemes liked CEZW are explored in section 4.8. The CEZW is a modified EZW for colour images proposed by Shen [1997]. The

CEZW used a modified SOT to exploit the correlation between the colour components of the YUV colour space. The results obtained using CEZW show that the PSNR from CEZW is higher than that from SPIHT at different data rates. The performance of the SOT from the CEZW algorithm was further evaluated by Saenz et al [1999] as part of an investigation into colour embedded image compression.

Work related to colour image coding prior to wavelet-based compression schemes are explored in sections 4.6 and 4.7. Section 4.6 explored a method called plateau coding for coding chromatic component of colour television signals [Limb and Rubinstein, 1974]. Section 4.7 touch on the application of subband coding to colour images [Gharavi and Tabatabai, 1988].

Lastly, the use of multiwavelets in colour image compression is discussed in section 4.9. A study by Rout and Bell [2002] shows that scalar wavelets like the $\mathbf{B}_{22/14}$ perform better than the balanced multiwavelets.

CHAPTER 5

Statistical and Frequency Properties of Colour Images

5.1. Introduction

The output of the wavelet filter is influenced by the input characteristics of the image (as discussed in section 1.2). As one of the main concerns of this research is colour images, this chapter will report a study that will look into the characteristics of colour images and how they influence the coding performance. Another concern of this research is to establish, which wavelet filter is best suited to compress different types of images. Hence, a study is conducted and reported in section 5.4 to address this concern.

5.2. Background on Colour

The human eye has three types of colour photoreceptor cells called cones and the ability of the human eye to perceive colour arises from the sensitivities of these cones. Each type of cone responds to a range of wavelengths. Owing to this property of the human visual system, all colours can be modelled as combinations of the three primary colour components that is, red, green, and blue (RGB). The Commission International de l'Eclairage - the International Commission on Illumination (CIE) have designated the following wavelength values for the three primary colours that is, red = 700nm, blue = 435.8nm and green = 546.1nm (see Figure 5.1). The relative amounts of these three primary colours needed to produce a colour of a given wavelength are called tristimulus values.

Colour images displayed on a computer monitor are in the RGB colour space, thus the colour images used in this research are analysed in the RGB colour space. The same images are also analysed in the YC_bC_r colour spaces. This is done because the YC_bC_r colour space is often used in image compression schemes (see section 5.3.2).

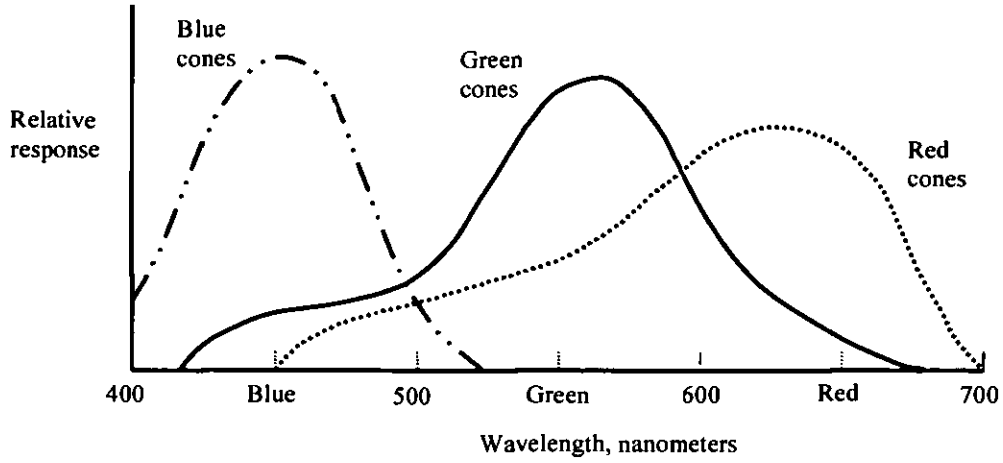


Figure 5.1 Wavelength values for the three primary colours

5.3. Colour Space

The colours perceived by the human eye can be represented by a colour space or colour model. Weighted combinations of stimuli at three wavelengths are adequate to describe all the colours the human eye perceives. These wavelengths can be used to form a coordinate system, from which the colour measurement process can be described. Although, a common colour space used in digital imagery is the RGB colour space, there are other colour spaces: the luminance/chrominance colour space; cyan, yellow, magenta (CYM) colour space; and hue, intensity saturation (HIS).

In this research as mentioned earlier the two colour spaces used are RGB and the luminance/chrominance. However, there are different variation of luminance/chrominance colour spaces, for example, YUV and YIQ, which are used in television broadcasting and the YC_bC_r colour space, which is used in most image compression schemes. Therefore, the following discussions will focus on the RGB and YC_bC_r colour spaces which are relevant to this research.

5.3.1. RGB Colour Space

A computer colour monitor displays colour images as a collection of pixels, which consist of three tiny dots in its spectral components of red, green, and blue. The RGB colour space is a three-dimensional orthogonal coordinate system (see Figure 5.2). The red, green and

blue intensity starts at zero at the origin and increases along the respective axes. The maximum value of each colour is 255 for an 8-bit depth and the net result is a cube.

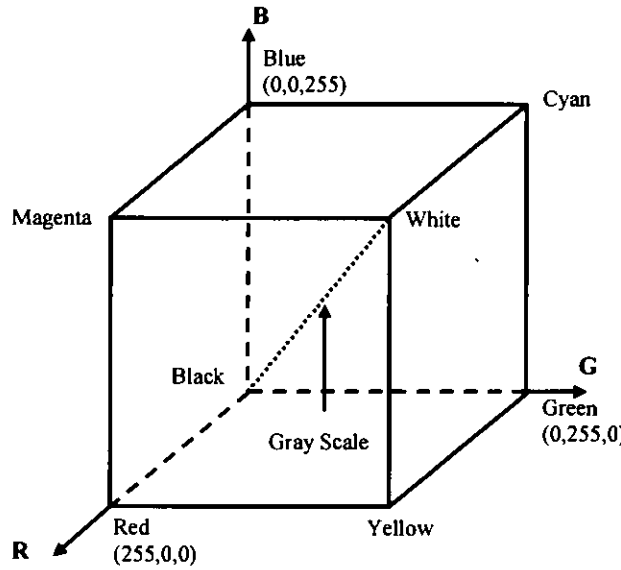


Figure 5.2 RGB colour space

Colour images in the RGB colour space are highly correlated between the components [Pratt, 1991; Gharavi and Tabatabai, 1988]. This correlation is often exploited in colour image compression schemes, for example, Shen, 1997 and Lervik and Ramstad, 1995.

5.3.2. YC_bC_r Colour Space

The YC_bC_r colour space is defined by the International Radio Consultative Committee, and is mainly used in the digital video paradigm and image compression schemes. It consists of the luminance (Y) and chrominance (C_bC_r), where Y is the light intensity, and C_b and C_r are the colour difference signals [Umbaugh, 1998, pp.31]. Pratt [1971] cited that the YC_bC_r colour space is a preferable colour space in transform coding and it is frequently used in image compression schemes [Lervik and Ramstad, 1995; Saenz et al, 1999]. This colour space is derived from the fact that the human vision perceives a colour stimulus in terms of luminance and chrominance attributes, rather than in terms of R, G and B values. The relation between the YC_bC_r space and the RGB space is defined by equation 3.5 in section 3.9.2.

Human vision is more sensitive to changes in the luminance than changes in chrominance and the YC_bC_r space exploits this advantage in a lossy compression scheme by quantizing the C_bC_r components more to yield higher compression ratios.

5.4. Variation in Coding Performance

To investigate the variation in coding performance, different wavelets were used to compress a selected set of colour images. The initial results show that different wavelets produce varied coding performance. The coding performance was compared with reference to peak signal-to-noise ratio (PSNR), compression ratio (CR) and bits-per-pixel (bpp). The results also indicate that PSNR, CR and bpp values are image dependent, that is the values obtained depend on certain characteristics of the image being compressed.

5.4.1. Image Classification

As mentioned in section 5.4, a selected set of colour images were used to investigate variation in coding performance. At this stage it was felt that different types of images have different characteristics that may influence coding performance. To verify this hypothesis the images were classified into various categories.

Some researchers have attempted to categorise images, such as Smith and Chang, [1997] and Gevers et al, [2000]. The main concern of Smith and Chang's [1997] work was to classify images for indexing and search capabilities of Web image search engines, and Gevers et al [2000] study dealt with the classification of images using both textual and image features.

This research initially attempted to classify colour images based on the classifications proposed by both Smith and Chang [1997] and Gevers et al [2000] but found them to be unsuitable. Drawing on the observations made by these researchers, this research classifies the images as follows:

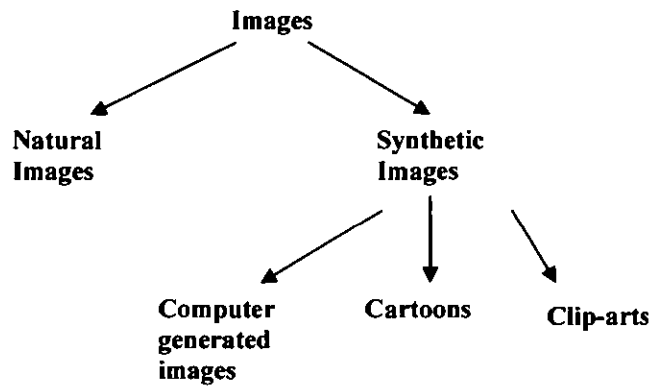


Figure5.3 Image Classification

The term 'synthetic image' referred to in this thesis is an image that is generated by a computer, a drawing (cartoon), or a clip-art.

Synthetic images differ from natural images in that synthetic images have sharp artificial colour transitions. On the other hand, natural images have gradual colour transition, for example photographs. Furthermore, sharp transition occurs between two different regions of constant colour in a synthetic image. In contrast, natural images do not have sharp edges like synthetic images instead the edges are often blurred. Also synthetic images have fewer colours than natural images. Cartoons and clip-arts are basically line drawings and therefore will have very well defined edges as compared to natural images.

5.4.2. Wavelet Filters Used

Five wavelets filters, `haar`, `db2`, `bior4.4`, `coif4`, and `sym4` from the Matlab wavelet toolbox are used. These are chosen because they are popular wavelets and are most frequently referred to in the literature, example Daubechies [1992], Burrus [1998], and Strang and Nguyen [1997]. The following is a brief explanation of these wavelets used in this work.

The wavelet names 'db*N*', 'bior*N*r.*N*d', 'coif*N*', and 'sym*N*' are short names for daubechies, biorthogonal, coiflets and symlets respectively. The *N*, *N*r, and *N*d are the order and they are integers. *N* is known as the length of support.

The wavelets used have different properties which are important in image compression. Some of these properties are compact support, symmetry, number of vanishing moments, and smoothness or regularity. These properties affect coding performance and therefore warrant further discussion. These properties are summarized in Table 5.1 for five of the wavelet families used in this study.

Symmetric or antisymmetric wavelets give rise to linear-phase filters. The usage of linear-phase filters in subband coding has been said to be desirable [Antonini et al, 1992]. The reason being that without linear-phase the phase distortion around edges is very visible. In addition for computational purposes, linear phase is more convenient because of the symmetry of the filters [Vetterli and Kovacevic, 1995, pp.413].

Table 5.1 Wavelet Properties [Misiti, 1997]

	haar	**db**	**Bior**	**coif**	**sym**
Orthogonal	Yes	Yes	No	Yes	Yes
Biorthogonal	Yes	Yes	Yes	Yes	Yes
Compact support	Yes	Yes	Yes	Yes	Yes
Support width	1	2N-1	2Nr+1 for rec., 2Nd+1 for dec.	6N-1	2N-1
Filters length	2	2N	max(2Nr,2Nd)+2	6N	2N
Regularity	Not continuous	about 0.2 N for large N	Nr-1 and Nr-2 at the knots	Information not available	Information not available
Symmetry	Yes	Not symmetric	Yes	Near symmetric	Near symmetric
No. of vanishing moments for psi (ψ)	1	N	Nr-1	2N	N

In terms of image coding performance, regularity is also desirable since a high regularity can give a better compression ratio. A wavelet filter is called regular if the filter has a certain number of zeros at the aliasing frequency and if it iterates towards a continuous function [Vetterli and Kovacevic, 1995, pp.414]. High regularity requires long filters but long filters can cause artifacts, for example ringing around edges [Vetterli and Kovacevic, 1995, pp.413].

Another important wavelet property is the number of 'p' vanishing moments. The 'p' moment of a wavelet is defined as:

$$\int x^p \tilde{\psi}(x) dx = 0$$

where $\tilde{\psi}(x)$ is the analysis wavelet, x^p is the signal or function, and $p = 0, \ldots, n$. The function x^p can be expressed by a polynomial. If a wavelet with enough number of vanishing moments, p, is used to analyze a polynomial with a degree less than p, then all detail coefficients will be zero [Unser, 2003; Strang and Nguyen, 1997, pp.227-228]. These zero coefficients can then be coded with fewer bits resulting in a higher compression ratio.

5.4.3. Results

The principal wavelet types supported by the Matlab wavelet toolbox were applied to a sample of sixty images. Based on the findings of Adams and Ward [2001] a three level decomposition is chosen. Adams and Ward [2001] show that coding efficiency is contributed by the first five decomposition levels. Based on this, it would be reasonable to use a three level decomposition. The performance of the image coder is evaluated based on the PSNR obtained. Some of the results are presented in Table 5.2. A full set of results can be found in Appendix 1.

The standard peak signal-to-noise ratio PSNR is defined by [Strang and Nguyen, 1997]:

$$PSNR = 10 * \log_{10}\left(255^2 / MSE\right) \tag{5.1}$$

and the mean square error (MSE) is defined by [Strang and Nguyen, 1997]:

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y) - \hat{I}(x,y)\right]^2 \tag{5.2}$$

where M and N are the row and column of the image respectively. I is the original image and $\hat{I}$ is the compressed image.

However, the PSNR of a colour image with red, green and blue components is defined by [Saenz et al, 1999]

$$\text{PSNR} = 10 * \log_{10}\left\{\frac{255^2}{\frac{MSE(red) + MSE(green) + MSE(blue)}{3}}\right\} \tag{5.3}$$

A high value of PSNR is good because it means that the signal-to-noise ratio is high which means less error in the image. In image compression, the 'signal' is the original image, and the 'noise' is the error in reconstruction.

Image compression involves reducing the size of image data, while retaining necessary information. The reduced file is known as the compressed file. Compression ratio is defined by [Umbaugh, 1998, pp.237]:

$$Compression\ ratio = \frac{Uncompressed\ file\ size}{Compressed\ file\ size}$$

An alternative way to state the compression is to use the terminology bits-per-pixel, which is defined by [Umbaugh, 1998, pp.238]:

$$Bits - per - pixel = \frac{No.of\ bits}{No.of\ pixels} = \frac{(8)(no.of\ bytes)}{Size\ of\ image}$$

Table 5.2 Initial results

Image	haar			db2			bior4.4			bior6.8			coif4			sym4		
	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp
peppers	30.28	13.40	0.597	30.90	***14.57***	0.549	31.29	12.98	0.616	31.32	9.92	0.807	31.23	8.16	0.980	31.23	13.06	0.613
bird	36.66	***45.18***	0.177	36.98	44.66	0.179	37.23	41.57	0.192	37.26	32.67	0.245	37.01	28.09	0.285	***37.29***	40.90	0.196
lenacolor	30.04	16.08	0.497	30.45	***17.54***	0.456	30.64	17.08	0.468	30.75	13.60	0.588	30.58	11.56	0.692	30.61	16.62	0.481
palmblades	26.67	6.72	1.190	26.96	***7.25***	1.103	27.27	7.21	1.109	***27.40***	6.06	1.320	27.25	5.34	1.498	27.19	7.10	1.127
snowwhite02	32.08	21.64	0.370	32.64	***22.70***	0.352	32.70	21.97	0.364	***33.14***	16.99	0.471	32.73	14.47	0.553	32.72	21.44	0.373
hercules01_256	30.63	12.09	0.662	30.60	***12.78***	0.627	30.78	12.68	0.631	30.72	10.31	0.776	30.51	8.98	0.891	30.73	12.47	0.641
toystory01_256	30.59	14.20	0.563	30.49	***14.97***	0.534	30.68	13.97	0.573	30.69	11.44	0.699	30.51	9.88	0.809	30.58	14.14	0.809
mermaid01_256	29.74	9.39	0.852	29.67	***9.83***	0.814	29.81	9.33	0.858	***29.82***	7.57	1.057	29.68	6.40	1.250	29.71	9.30	0.860
clipart09_256	37.13	***18.68***	0.480	33.02	11.27	0.710	31.80	10.84	0.738	30.63	7.62	1.050	30.54	6.30	1.271	30.98	10.63	0.752
clipart06_256	41.34	***43.28***	0.185	36.07	32.58	0.246	35.81	29.21	0.274	35.38	24.13	0.332	34.54	20.17	0.397	35.78	29.44	0.272
clipart10_256	37.18	***24.21***	0.330	34.10	18.79	0.426	33.65	17.48	0.458	32.88	15.59	0.513	32.79	13.54	0.591	32.99	17.61	0.454
clipart05_256	35.64	***18.18***	0.494	32.65	14.26	0.561	32.17	13.74	0.582	31.77	11.30	0.708	31.41	9.92	0.806	32.07	13.81	0.579

It can be seen that different wavelet filters used on the same image produce different PSNR, compression ratio (CR), and bpp values. This is consistent with an earlier work by Yap and Rahman, [2003a]. For example, the PSNR value varies from 37.13 to 30.63 dB for the '`clipart09_256`' image using the six different wavelets as shown in Table 5.1. The results also show that some wavelet filters performed better than others in terms of PSNR, for example for '`clipart09_256`' the '`haar`' wavelet filter performed the best. However, the results also show that the same wavelet filter does not necessarily yield the best PSNR and best CR. For example, the PSNR for the '`peppers`' image is 31.32 dB using the '`bior6.8`' wavelet filter but the '`db2`' wavelet filter yields a better CR. Although the '`bior6.8`' wavelet filter slightly outperforms the '`bior4.4`' wavelet filter in terms of PSNR, overall the best combination of PSNR and CR is produced by the bior4.4 (9/7 tap) wavelet. This result is wholly consistent with the selection of the Daubechies 9/7 tap wavelet as the basis of the JPEG2000 standard. For clipart images, the '`haar`' wavelet

outperforms all the other wavelets and therefore it is the preferred wavelet for such images. However, there is a significant variation of PSNR values for different clipart images when the images are compressed using the 'haar' wavelet.

Furthermore, some wavelet filters perform better than others depending on the image being compressed; for example the 'bior4.4' wavelet performs better than the 'haar' wavelet for the 'peppers' image in terms of PSNR. It is also observed that using biorthogonal wavelets, that is 'bior4.4' and 'bior6.8', to compress natural and synthetic images yields the best PSNR. This result is consistent with the findings of Rout and Bell [2002] that cited biorthogonal wavelets provide the best performance in image compression.

The variation in PSNR may be attributed to the characteristics of the image and this is consistent with the study of Saha and Vemuri [1999a]. Any of the image characteristics could have caused these variations for example, the number of colours or the distinct edges of an image.

At this stage the results show that the coding performance measures (PSNR, CR and bpp) differ depending on both the type of wavelet filters used and the characteristics of the images being coded. The next section discusses the work carried out to establish why the coding performance measures vary from image to image.

5.5. Image Statistics

As mentioned in section 5.1, one of the concerns of this research is to investigate and establish which characteristics and statistical features of a colour image influence the coding performance. To explore this problem and to understand how the various image features affects the coding performance, grey-level image histogram statistics are used to analyse colour images. The grey-level image histogram statistics present a statistical analysis of an image that can provide useful information about the characteristics of the image. As mentioned in chapter one, grey-scale image statistical analysis is being used because there is limited data on coding performance measures for colour images. The data gathered allows certain quantitative and qualitative comparisons to be made. The common statistical features are mean, standard deviation, variance, skewness, kurtosis, and entropy [Saha and Vemuri, 2000a; Umbaugh, 1998]. In addition, image gradient is also used [Saha

and Vemuri, 2000c]. Frequency characteristics of the images such as spectral frequency measure (SFM) and spatial frequency (SF) are also explored. The following sections will give a brief explanation of the image statistics used in this research.

5.5.1. Mean

This is an average, which indicates the general brightness of the image and is given by:

$$mean = \sum I(x, y)/(x * y) \tag{5.4}$$

where $\sum I(x, y)$ is the summation of all pixel values of the image and $(x * y)$ is the size of the image [Burdick, 2000, pp.37]. An image with a high mean indicates that the image is a bright image and a dark image will have a low mean.

5.5.2. Standard deviation

The measure of the frequency distribution of pixel values of an image is known as the standard deviation of that image. The standard deviation (Std. Dev.) can be calculated by:

$$\textbf{Std. Dev} = \sqrt{\left(\sum {I_{x,y}}^2 /(x * y) - (mean)^2\right)} \tag{5.5}$$

where $\sum {I_{x,y}}^2$ is the sum of the squares of all pixel values of the image [Burdick, 2000, pp.37].

5.5.3. Variance

The variance is the square of the standard deviation and is calculated using [Parker, 1997, pp.120]:

$$Variance(\sigma) = (Std.Dev)^2 \tag{5.6}$$

An image with a high variance means that the image has a high contrast, and an image with a low variance indicates that the image has a low contrast.

5.5.4. Skewness

Skewness is a measure of symmetry, or more precisely, the lack of symmetry of a histogram. A distribution or data set is symmetric, if it looks the same to the left and right of the centre point. Skewness is calculated by using:

$$s = \frac{1}{N}\sum\left[\frac{I_{x,y} - mean}{\sigma}\right]^3 \tag{5.7}$$

where N is the number of pixels [Parker, 1997, pp.153].

5.5.5. Kurtosis

Kurtosis is a measure of whether the data is peaked or flat relative to a normal distribution. Data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. Kurtosis is calculated by using:

$$k = \frac{1}{N}\sum\left[\frac{I_{x,y} - mean}{\sigma}\right]^4 - 3 \tag{5.8}$$

where N is the number of pixels [Parker, 1997, pp.153].

5.5.6. Entropy

Assuming an information source with n possible symbols x and that symbol i will occur with probability $p(x_i)$ then the entropy associated with the source of the symbols X is defined, in general, as:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log(p(x_i)) \tag{5.9}$$

where entropy is measured in bits/symbol [Parker, 1997, pp.121].

If an image is thought of as a source of symbols, or grey levels, then entropy is the measure of information content [Parker, 1997, pp.120] of an image. The entropy of an image in this research is measured by:

a. zero-order entropy,

$$H(0) = \log_2 N_i \tag{5.10}$$

where N_i is the different number of pixels. The zero order entropy is $\log_2$ of different values of pixels or average number of bits required to represent one pixel [Grigorescu et al, 2003; Sprljan et al, 2003].

b. first-order entropy,

$$H(1) = -\sum_{i=1}^{n} P_i \log_2 P_i \qquad \text{bits/pixel} \qquad \text{[Parker,1997, pp.121]} \tag{5.11}$$

c. second-order entropy,

$$H(2) = -p(i,j) * \log_2 p(i,j) \tag{5.12}$$

where $p(i,j)$ denotes the probability that a pixel has value i while its neighbouring pixel has value j [Sprljan et al, 2003].

5.5.7. Image Statistics Results

Various image statistics for a set of sixty different colour images were calculated in the RGB and YC_bC_r colour space. The following tables show a sample of the results of image statistics in the RGB and YC_bC_r colour space respectively. A complete set of results can be found in Appendix 2.

Table 5.3 Image statistics in RGB colour space

	Mean			Std.Dev.			Variance			Skewness			Kurtosis		
Image	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
bird	108.17	115.48	127.46	18.77	19.02	22.00	352.17	361.94	483.92	2.80	0.11	-2.37	23.84	14.82	7.69
bulehills256	59.54	132.63	209.59	72.40	78.92	44.17	5241.80	6228.12	1951.36	0.79	0.24	-0.34	-0.93	-1.64	-1.47
columns	96.71	88.54	94.61	62.61	63.40	70.45	3919.84	4019.35	4963.07	0.64	0.60	0.88	-0.49	-0.68	-0.33
snowwhite02	83.65	79.07	74.57	43.15	40.87	35.65	1861.82	1670.58	1271.24	0.90	0.91	0.89	0.59	0.56	1.20
toystory02_256	165.37	151.35	117.14	81.04	85.87	68.87	6567.99	7373.49	4743.63	-0.36	-0.15	0.23	-1.53	-1.75	-1.72
toystory01_256	78.87	84.85	80.52	58.05	57.81	57.27	3369.66	3342.08	3280.07	0.84	0.68	0.75	0.04	-0.17	0.18
clipart09_256	188.72	105.24	154.94	111.84	125.54	124.51	12508.45	15761.14	15503.57	-1.10	0.36	-0.44	-0.80	-1.87	-1.81
clipart06_256	196.88	197.35	197.46	104.57	104.23	104.18	10935.57	10864.24	10854.24	-1.31	-1.32	-1.33	-0.22	-0.19	-0.18
clipart10_256	191.84	181.45	179.19	76.96	79.64	80.42	5923.45	6342.21	6467.92	-1.48	-0.99	-0.89	1.49	0.32	0.09

Table 5.4 Image statistics in YC_bC_r colour space

	Mean			Std.Dev.			Variance			Skewness			Kurtosis		
Image	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
bird	0.450	0.028	-0.018	0.073	0.024	0.020	0.005	0.001	0.000	0.377	-3.473	5.305	15.619	13.624	49.911
bulehills256	0.469	0.199	-0.168	0.281	0.079	0.052	0.079	0.006	0.003	0.366	-0.720	-0.042	-1.529	-0.867	-0.952
columns	0.360	0.007	0.014	0.250	0.025	0.025	0.062	0.001	0.001	0.667	0.214	2.776	-0.569	0.508	9.963
snowwhite02	0.313	-0.012	0.010	0.160	0.021	0.017	0.026	0.000	0.000	0.904	-0.834	1.070	0.613	1.206	4.415
toystory02_256	0.595	-0.076	0.038	0.315	0.076	0.066	0.099	0.006	0.004	-0.157	-0.683	1.399	-1.697	0.380	1.521
toystory01_256	0.324	-0.005	-0.010	0.224	0.039	0.034	0.050	0.002	0.001	0.735	-0.265	0.255	-0.031	0.614	1.143
clipart09_256	0.533	0.042	0.148	0.392	0.272	0.249	0.154	0.074	0.062	0.240	0.668	0.665	-1.770	-0.469	-1.461
clipart06_256	0.773	0.001	-0.001	0.409	0.013	0.021	0.167	0.000	0.000	-1.319	27.852	-23.073	-0.193	868.022	539.302
clipart10_256	0.723	-0.011	0.021	0.309	0.013	0.023	0.095	0.000	0.001	-1.121	-0.194	0.194	0.625	-1.963	-1.963

To verify if there is any relationship between these image statistics and coding performance, that is, PSNR, CR, and bpp (obtained from section 5.4) the coefficient of correlation (r) is computed. The coefficient of correlation between the image statistic of each colour component and coding performance measure is computed using the following definition:

$$r = \frac{S_{xy}}{\sqrt{S_{xx} S_{yy}}} \tag{5.13}$$

where $S_{xy} = \frac{1}{n}\sum xy - \overline{xy}$, $S_{xx} = \frac{1}{n}\sum x^2 - \overline{x}^2$, and $S_{yy} = \frac{1}{n}\sum y^2 - \overline{y}^2$. x and y are the variables [Lewis and Traill, 1999, pp.167].

A correlation is a measure of the relation between two or more variables. The r between the image statistics and coding performance indicates the strength of the relationship between the variables. A strong r will indicate a strong relationship or connection in which one variable affects or depends on the other. A weak correlation will indicate the opposite.

The coefficient of correlation for both RGB and YC_bC_r images is presented in Table 5.5 and Table 5.6 respectively. In addition to this, the coefficient of determination (r^2) is also computed. The r^2 value indicates the proportion of common variation in the two variables, that is, the 'strength' of the relationship. This is important when evaluating the correlation between two variables.

As there is no defined method for determining whether a connection is weak or strong, for this research a weak connection is defined as having r^2 of less than 0.4. A scatter plot of PSNR and mean (red) show that most data points are scattered around the trend line and do not fall on the trend line (see Figure 5.4). This indicates a weak correlation between the two variables. Hence, based on this observation, this thesis argues that the image statistics used in section 5.5 cannot be used to explain the variation in the coding performance measures.

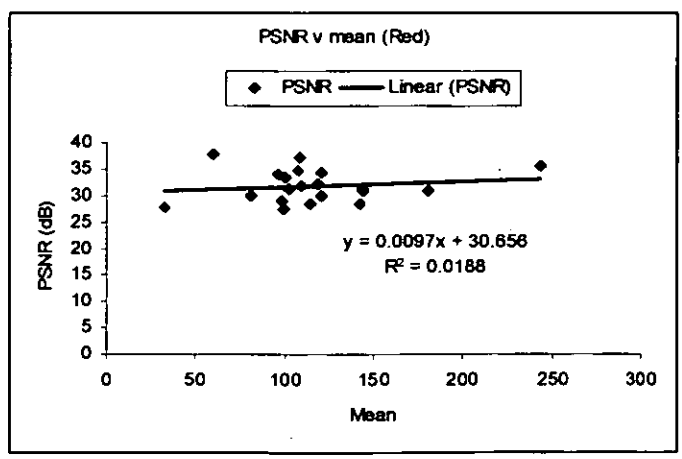


Figure 5.4 PSNR v mean (Red)

Table 5.5 Coefficient of correlation for RGB images

Natural Images

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
mean and PSNR	0.1372	0.0188	0.3135	0.0982	0.4812	0.2316
Std.Dev. and PSNR	-0.2440	0.0595	-0.0525	0.0028	0.0045	0.0000
Var. and PSNR	-0.1385	0.0192	0.0288	0.0008	0.0033	0.0000
Skew and PSNR	-0.0683	0.0047	-0.4086	0.1670	-0.5812	0.3149
Kurt. and PSNR	0.4646	0.2159	0.3349	0.1121	-0.2468	0.0609
mean and CR	0.0240	0.0005	0.2719	0.0739	0.5413	0.2930
Std.Dev. and CR	-0.2884	0.0832	-0.1605	0.0258	-0.0662	0.0044
Var. and CR	-0.1751	0.0307	-0.0647	0.0042	-0.0452	0.0020
Skew and CR	0.0950	0.0090	-0.2899	0.0840	-0.5821	0.3389
Kurt. and CR	0.5700	0.3250	0.4712	0.2220	-0.1789	0.0320
mean and bpp	-0.1188	0.0141	-0.1457	0.0212	-0.4724	0.2231
Std.Dev. and bpp	0.1749	0.0306	0.0765	0.0059	-0.0929	0.0088
Var. and bpp	0.0888	0.0079	0.0209	0.0004	-0.0777	0.0060
Skew and bpp	0.0520	0.0027	0.2725	0.0742	0.6415	0.4115
Kurt. and bpp	-0.3726	0.1389	-0.2801	0.0785	0.4759	0.2265

Synthetic images

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
mean and PSNR	-0.1518	0.0230	-0.3381	0.1130	-0.2616	0.0685
Std.Dev. and PSNR	-0.5938	0.3526	-0.3563	0.1270	-0.1825	0.0333
Var. and PSNR	-0.5666	0.3210	-0.3352	0.1124	-0.1888	0.0356
Skew and PSNR	0.1664	0.0277	0.0987	0.0097	-0.0192	0.0004
Kurt. and PSNR	0.3710	0.1377	-0.0237	0.0006	-0.1958	0.0383
mean and CR	-0.1720	0.0296	-0.2863	0.0820	-0.2863	0.0820
Std.Dev. and CR	-0.5758	0.3315	-0.3932	0.1546	-0.3888	0.1511
Var. and CR	-0.5767	0.3325	-0.3882	0.1507	-0.4021	0.1617
Skew and CR	0.3220	0.1037	0.2603	0.0677	-0.0332	0.0011
Kurt. and CR	0.3331	0.1110	0.0516	0.0027	-0.1387	0.0192
mean and bpp	0.0805	0.0065	0.2634	0.0694	0.1124	0.0126
Std.Dev. and bpp	0.6241	0.3895	0.3511	0.1233	0.3444	0.1186
Var. and bpp	0.6321	0.3995	0.3398	0.1154	0.3645	0.1329
Skew and bpp	-0.2521	0.0636	-0.2034	0.0414	0.0701	0.0049
Kurt. and bpp	-0.3336	0.1113	0.0371	0.0014	0.1619	0.0262

Clipart images

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
mean and PSNR	0.1558	0.0243	-0.2186	0.0478	-0.1055	0.0111
Std.Dev. and PSNR	-0.5352	0.2865	-0.2302	0.0530	-0.4077	0.1662
Var. and PSNR	-0.4485	0.2012	-0.2252	0.0507	-0.4208	0.1771
Skew and PSNR	-0.1605	0.0258	0.2336	0.0546	0.1659	0.0275
Kurt. and PSNR	0.2705	0.0732	0.3753	0.1408	0.2664	0.0710
mean and CR	0.2027	0.0411	0.2287	0.0523	0.2430	0.0591
Std.Dev. and CR	-0.5943	0.3532	-0.2456	0.0603	-0.2743	0.0753
Var. and CR	-0.4583	0.2100	-0.2767	0.0766	-0.3194	0.1020
Skew and CR	0.1427	0.0204	-0.2210	0.0488	-0.1872	0.0350
Kurt. and CR	-0.0601	0.0036	0.3355	0.1126	0.0808	0.0065
mean and bpp	-0.0276	0.0008	-0.0360	0.0013	-0.0827	0.0068
Std.Dev. and bpp	0.4863	0.2365	0.3882	0.1507	0.4358	0.1899
Var. and bpp	0.4330	0.1875	0.4008	0.1606	0.4653	0.2165
Skew and bpp	0.0010	0.0000	-0.0096	0.0001	-0.0009	0.0000
Kurt. and bpp	-0.0969	0.0094	-0.4248	0.1805	-0.1907	0.0364

Table 5.6 coefficient of correlation for YC_bC_r images

Natural images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
mean and PSNR	0.3173	0.1007	0.3291	0.1083	-0.1624	0.0264
Std.Dev. and PSNR	-0.1300	0.0169	-0.0292	0.0009	0.1490	0.0222
Var. and PSNR	-0.0483	0.0023	0.0345	0.0012	0.1480	0.0219
Skew and PSNR	-0.3388	0.1148	-0.4228	0.1787	0.1736	0.0301
Kurt. and PSNR	0.3417	0.1168	0.1377	0.0190	0.2345	0.0550
mean and CR	0.2564	0.0657	0.4540	0.2061	-0.2878	0.0828
Std.Dev. and CR	-0.1936	0.0375	-0.0578	0.0033	-0.0252	0.0006
Var. and CR	-0.0953	0.0091	-0.0020	0.0000	-0.0249	0.0006
Skew and CR	-0.2025	0.0410	-0.4304	0.1852	0.2919	0.0852
Kurt. and CR	0.4785	0.2289	0.2314	0.0535	0.3209	0.1030
mean and bpp	-0.2123	0.0451	-0.4057	0.1646	0.0602	0.0036
Std.Dev. and bpp	0.0962	0.0093	0.1356	0.0184	-0.1076	0.0116
Var. and bpp	0.0349	0.0012	0.0919	0.0084	-0.0748	0.0056
Skew and bpp	0.2241	0.0502	0.3292	0.1083	-0.0324	0.0010
Kurt. and bpp	-0.2766	0.0765	-0.0921	0.0085	-0.0494	0.0024

Synthetic images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
mean and PSNR	-0.2908	0.0846	-0.0187	0.0003	0.2339	0.0547
Std.Dev. and PSNR	-0.4242	0.1800	-0.2548	0.0649	-0.2774	0.0770
Var. and PSNR	-0.3875	0.1502	-0.2584	0.0668	-0.2962	0.0877
Skew and PSNR	0.0752	0.0057	0.1033	0.0107	0.2976	0.0886
Kurt. and PSNR	0.0914	0.0084	-0.0779	0.0061	0.2202	0.0485
mean and CR	-0.2522	0.0636	0.1411	0.0199	0.1227	0.0151
Std.Dev. and CR	-0.4037	0.1630	-0.4327	0.1872	-0.5725	0.3278
Var. and CR	-0.3887	0.1511	-0.4029	0.1623	-0.5613	0.3151
Skew and CR	0.2620	0.0686	0.0099	0.0001	0.2713	0.0736
Kurt. and CR	0.1268	0.0161	0.0565	0.0032	0.3959	0.1567
mean and bpp	0.2124	0.0451	0.2671	0.0714	-0.0549	0.0030
Std.Dev. and bpp	-0.0504	0.0025	0.3166	0.1002	0.1378	0.0189
Var. and bpp	-0.0984	0.0097	0.3098	0.0960	0.0917	0.0084
Skew and bpp	-0.3163	0.1001	0.1105	0.0122	-0.0821	0.0067
Kurt. and bpp	-0.1153	0.0133	-0.2268	0.0514	-0.2474	0.0612

Clipart images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
mean and PSNR	-0.1190	0.0142	0.0077	0.0001	0.3875	0.1502
Std.Dev. and PSNR	-0.4115	0.1693	-0.4266	0.1819	0.2478	0.0614
Var. and PSNR	-0.4086	0.1670	-0.4221	0.1782	0.2117	0.0448
Skew and PSNR	0.1425	0.0203	0.2070	0.0428	-0.1978	0.0391
Kurt. and PSNR	0.3663	0.1342	0.2624	0.0688	0.2506	0.0628
mean and CR	0.2528	0.0639	0.0102	0.0001	-0.0625	0.0039
Std.Dev. and CR	-0.3479	0.1210	-0.4930	0.2431	-0.0384	0.0015
Var. and CR	-0.3425	0.1173	-0.4536	0.2057	-0.0850	0.0072
Skew and CR	-0.3178	0.1010	0.2093	0.0438	-0.2076	0.0431
Kurt. and CR	0.4427	0.1960	0.2849	0.0812	0.2789	0.0778
mean and bpp	0.0589	0.0035	-0.3983	0.1587	0.1884	0.0355
Std.Dev. and bpp	0.2534	0.0642	-0.0534	0.0028	-0.0671	0.0045
Var. and bpp	0.2571	0.0661	-0.1287	0.0166	-0.1043	0.0109
Skew and bpp	0.0016	0.0000	-0.2301	0.0529	0.1256	0.0158
Kurt. and bpp	-0.1642	0.0269	-0.2206	0.0487	-0.1604	0.0257

As can be seen from the Table 5.5 and Table 5.6, all the image statistics have only a weak connection with the coding performance measures. This is consistent with an earlier work by Yap and Rahman, [2003b].

In addition to the mentioned image statistics, the three entropy measures, that is, *H(0), H(1), H(2),* are also computed in the RGB and YC_bC_r colour space. A sample of these results is presented in the following tables.

Table 5.7 Entropy measures in RGB colour space

	Red			Green			Blue		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
bird	7.8580	4.9019	4.0235	7.8455	4.8410	3.9642	7.8202	4.9375	4.3459
bulehills256	7.8704	6.0232	4.6470	7.8642	7.2334	5.4681	7.0768	5.6306	4.4149
columns	7.9580	7.4940	5.8346	7.7879	7.4453	5.7333	7.9715	7.5467	5.9382
snowwhite02	7.9425	7.2809	6.1975	7.9484	7.1898	6.1461	8.0000	7.0880	6.0953
toystory02_256	7.9009	7.0150	5.6206	7.9069	6.6024	5.3025	7.8455	6.7185	5.4353
toystory01_256	8.0000	7.4877	6.3619	8.0000	7.5870	6.4021	8.0000	7.5238	6.4073
clipart09_256	1.0000	0.8266	0.5079	1.0000	0.9778	0.6291	1.0000	0.9663	0.6340
clipart06_256	1.5850	0.9495	0.5418	2.0000	0.9522	0.5465	1.5850	0.9454	0.5428
clipart10_256	1.5850	1.3822	0.8077	1.5850	1.3822	0.8077	1.5850	1.3822	0.8077

Table 5.8 Entropy measures in YC_bC_r colour space

	Y			Cb			Cr		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
bird	7.8455	4.8210	6.5996	4.0000	2.7894	4.3005	6.2479	0.9426	0.8322
bulehills256	7.7879	7.1411	6.9843	6.2095	5.1897	5.2306	0.0000	0.0000	0.0000
columns	7.8329	7.4748	7.6922	4.9542	2.8885	3.6772	5.5236	3.4113	4.9402
snowwhite02	7.9366	7.1954	7.5736	3.4594	1.2520	1.5084	5.1293	3.0677	4.0263
toystory02_256	7.8580	6.6446	6.7628	4.8074	1.1272	1.0107	6.2668	4.2487	4.3022
toystory01_256	8.0000	7.5481	7.4985	5.3576	2.5959	2.6765	5.5850	2.1055	2.3149
clipart09_256	2.3219	1.9867	1.4171	1.0000	0.7778	0.5434	1.5850	1.1106	0.8471
clipart06_256	2.3219	0.8697	0.5570	1.5850	0.0252	0.0185	0.0000	0.0000	0.0000
clipart10_256	1.5850	1.3922	1.0416	0.0000	0.0000	0.0000	1.0000	0.9933	0.8201

As with the image statistics, r and r^2 are computed to verify if there is any relationship between these entropies and coding performance measures. The results are presented in Table 5.9.

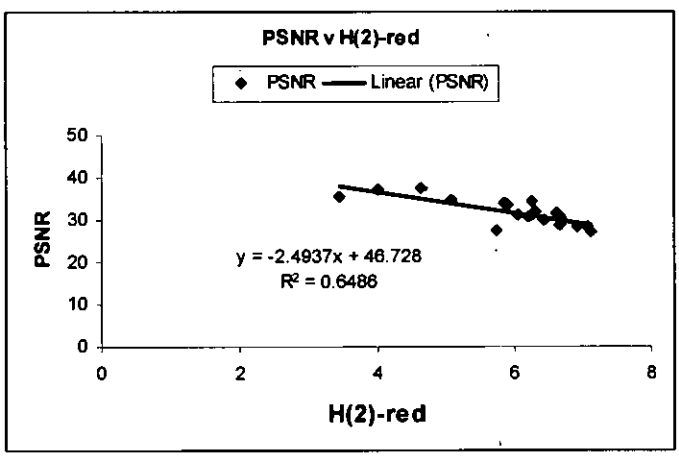
It can be seen from the results that most of the zero-order entropy measures in the RGB colour space have only a weak connection with the coding performance measures, except

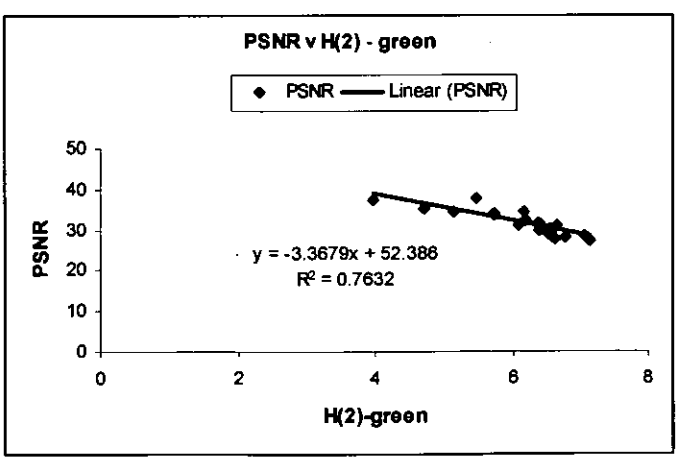
for the synthetic images. In the natural images category, there is a moderate negative correlation between *H(1)* and PSNR and CR. The results also show that there is a moderate positive correlation between *H(1)* and bpp. There is a also a strong correlation between *H(2)* and the coding measures in the red and green components but the correlation in the blue component is weak. For synthetic images, there is a strong negative correlation between *H(0)* and PSNR and CR in the red and green colour components. However, the correlation in the blue colour component is weak. To get a better picture, scatter plots in Figure 5.5 were made. By visual inspection it was decided that the straight line curve is the best for describing the overall pattern of the data. In this case the mathematical equation is of the form of a linear equation:

$$y = bx + a$$

where a is the y-intercept and b is the slope of the line.

It can be seen from Figure 5.5 that most of the data for the red and green component fall on the trend line, whereas for the blue component the data is scatted round about the trend line.





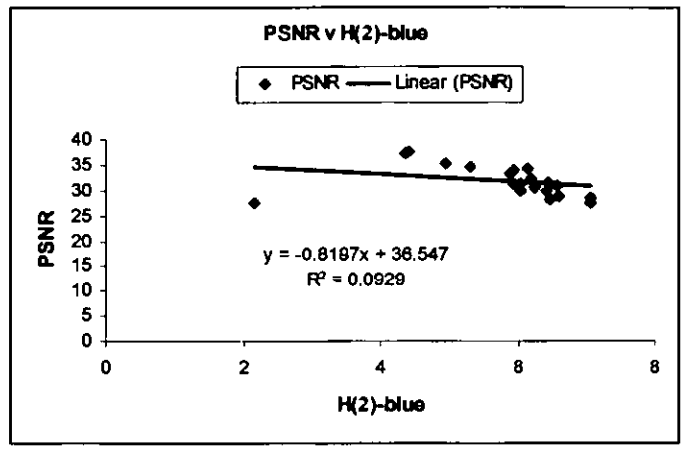


Figure 5.5 Scatter plots PSNR and H(2)

Table 5.9 Entropy correlation results for RGB images

Natural images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
H(0) and PSNR	-0.3323	0.1104	-0.3685	0.1358	-0.4109	0.1689
H(0) and CR	-0.3362	0.1130	-0.3443	0.1186	-0.4447	0.1978
H(0) and bpp	0.3512	0.1233	0.3946	0.1557	0.3586	0.1286
H(1) and PSNR	-0.5411	0.2928	-0.5180	0.2684	-0.0629	0.0040
H(1) and CR	-0.6361	0.4047	-0.6288	0.3954	-0.1514	0.0229
H(1) and bpp	0.4617	0.2132	0.4597	0.2113	-0.2211	0.0489
H(2) and PSNR	-0.8041	0.6466	-0.8738	0.7632	-0.3047	0.0929
H(2) and CR	-0.8393	0.7044	-0.8935	0.7984	-0.3480	0.1211
H(2) and bpp	0.7442	0.5538	0.8141	0.6627	0.0120	0.0001

Synthetic images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
H(0) and PSNR	-0.8183	0.6696	-0.8199	0.8722	-0.6439	0.4146
H(0) and CR	-0.7095	0.5033	-0.7005	0.4906	-0.5410	0.2927
H(0) and bpp	0.6186	0.3829	0.6487	0.4208	0.5154	0.2657
H(1) and PSNR	-0.1058	0.0112	-0.1120	0.0125	-0.1304	0.0170
H(1) and CR	-0.0588	0.0035	-0.0143	0.0002	0.0962	0.0093
H(1) and bpp	0.1111	0.0123	0.0395	0.0016	-0.0866	0.0075
H(2) and PSNR	-0.2372	0.0563	-0.2722	0.0741	-0.2817	0.0794
H(2) and CR	-0.0920	0.0085	-0.0934	0.0087	-0.0068	0.0000
H(2) and bpp	0.0948	0.0090	0.0902	0.0081	0.0009	0.0000

Clipart images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
H(0) and PSNR	-0.0967	0.0094	-0.0748	0.0056	-0.0653	0.0043
H(0) and CR	-0.0882	0.0078	-0.0178	0.0003	-0.0154	0.0002
H(0) and bpp	-0.1763	0.0311	-0.2082	0.0434	-0.2087	0.0435
H(1) and PSNR	-0.0759	0.0058	-0.1194	0.0143	0.0200	0.0004
H(1) and CR	-0.0875	0.0077	-0.0867	0.0075	-0.0134	0.0002
H(1) and bpp	-0.1227	0.0151	-0.0929	0.0086	-0.1767	0.0312
H(2) and PSNR	-0.1056	0.0112	-0.1627	0.0265	-0.0223	0.0005
H(2) and CR	-0.1053	0.0111	-0.1044	0.0109	-0.0396	0.0016
H(2) and bpp	-0.0995	0.0099	-0.0899	0.0081	-0.1467	0.0215

The results from Table 5.10 show that the zero-order entropy measure in the YC_bC_r colour space has only a weak connection with all the coding performance measures. The corresponding r^2 supports this observation. Therefore, it is concluded that the zero-order entropy in the YC_bC_r colour space cannot be used to explain the variation in coding performance measures.

Table 5.10 Entropy correlation results for YC_bC_r images

Natural imagee:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
H(0) and PSNR	-0.3715	0.1380	-0.1837	0.0337	-0.1771	0.0314
H(0) and CR	-0.3266	0.1067	-0.0448	0.0020	-0.2854	0.0814
H(0) and bpp	0.3540	0.1253	0.1148	0.0132	0.1400	0.0196
H(1) and PSNR	-0.5232	0.2738	0.0770	0.0059	-0.0797	0.0064
H(1) and CR	-0.6363	0.4049	0.2353	0.0554	-0.2515	0.0633
H(1) and bpp	0.4416	0.1950	-0.1481	0.0219	-0.0207	0.0004
H(2) and PSNR	-0.6384	0.4076	0.0802	0.0064	-0.1630	0.0266
H(2) and CR	-0.6810	0.4638	0.2546	0.0648	-0.2995	0.0897
H(2) and bpp	0.5269	0.2776	-0.1439	0.0207	0.0529	0.0028

Synthetic images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
H(0) and PSNR	0.0092	0.0001	-0.2086	0.0435	-0.2086	0.0435
H(0) and CR	0.1073	0.0115	-0.2515	0.0633	-0.0143	0.0002
H(0) and bpp	-0.0483	0.0023	0.2018	0.0407	-0.0441	0.0019
H(1) and PSNR	-0.1582	0.0250	-0.3733	0.1393	0.1418	0.0201
H(1) and CR	-0.0651	0.0042	-0.3236	0.1047	0.0632	0.0040
H(1) and bpp	0.1281	0.0164	0.3597	0.1294	-0.0610	0.0037
H(2) and PSNR	-0.0686	0.0047	-0.3387	0.1147	0.1533	0.0235
H(2) and CR	0.0127	0.0002	-0.2656	0.0705	0.1018	0.0104
H(2) and bpp	0.0298	0.0009	0.2992	0.0895	-0.0592	0.0035

Clipart images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
H(0) and PSNR	0.1045	0.0109	0.1966	0.0387	0.1788	0.0320
H(0) and CR	-0.0228	0.0005	0.0714	0.0051	0.0535	0.0029
H(0) and bpp	-0.0852	0.0073	-0.1342	0.0160	-0.2025	0.0410
H(1) and PSNR	0.1674	0.0280	0.2333	0.0544	0.1150	0.0132
H(1) and CR	0.0632	0.0040	0.1365	0.0186	0.0952	0.0091
H(1) and bpp	-0.2241	0.0502	-0.2285	0.0522	-0.1781	0.0317
H(2) and PSNR	0.1702	0.0290	0.2411	0.0581	0.0821	0.0067
H(2) and CR	0.0611	0.0037	0.1399	0.0196	0.0734	0.0054
H(2) and bpp	-0.2235	0.0499	-0.2412	0.0582	-0.1502	0.0225

5.6. Image Gradient

Image gradient is a measure of image edge strength and orientation. An edge is defined by a change in grey level. This change of grey level in an image is big near an edge and small in areas where the grey level is regular. Image gradient can be used to provide an indication of how busy the image is in terms of edges. Saha and Vemuri [2000b] defined image gradient as:

$$IG = \frac{1}{M * N}\left[\sum_{i=1}^{M-1}\sum_{j=1}^{N}\left|I(i,j) - I(i+1,j)\right| - \sum_{i=1}^{M}\sum_{j=1}^{N-1}\left|I(i,j) - I(i,j+1)\right|\right] \qquad (5.13)$$

5.6.1. Image Gradient Results

The image gradient measures for a set of sixty different colour images were calculated in the RGB and YC_bC_r colour space. The following table shows a sample of the results of image gradient measures.

Table 5.11 Image gradient measure results

	RGB			YCbCr		
Image	ImgGrad (R)	ImgGrad (G)	ImgGrad (B)	ImgGrad (Y)	ImgGrad (Cb)	ImgGrad (Cr)
bird	4.50	4.47	6.40	4.08	2.63	1.69
bulehills256	4.39	4.97	4.27	4.57	0.81	1.13
columns	9.16	8.64	9.75	8.54	2.25	2.16
couple	8.70	8.26	10.18	8.11	2.96	2.46
snowwhite02	13.17	13.04	12.88	12.95	0.95	0.81
toystory02_256	15.51	13.71	13.13	13.56	3.41	3.40
toystory01_256	20.21	20.06	20.41	19.78	2.95	2.73
mermaid01_256	27.91	25.65	25.11	24.53	5.55	7.85
clipart09_256	10.61	17.28	18.79	14.61	5.75	5.90
clipart06_256	6.71	7.01	7.15	6.94	0.12	0.16
clipart10_256	12.92	11.85	11.62	12.13	0.30	0.55
clipart05_256	6.54	8.46	9.05	7.88	1.85	2.45

The r and r^2 values are computed to verify if there is any relationship between image gradient and coding performance measures. The results are presented in Table 5.12. The results indicate there is a strong negative correlation between the image gradient and PSNR, and between image gradient and CR for natural and synthetic images. The results also indicate that there is a strong positive correlation between image gradient and bpp. However, a closer examination reveals that the correlation for the blue colour component is

not as strong as the correlation for red and green colour components. As for the clipart images there is a positive correlation between image gradient and all three coding performance measures for the red and green colour components. The results also show that there is a weaker correlation for the blue colour component.

In addition to the r and r^2 computations, the scatter plots are made to see the strength of the correlation between image gradient and PSNR (Figure 5.6).

Table 5.12 Image gradient correlation results for RGB images

Natural images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
ImgGrad and PSNR	-0.9254	0.8564	-0.9421	0.8876	-0.7530	0.5671
ImgGrad and CR	-0.8388	0.7037	-0.8553	0.7315	-0.6744	0.4548
ImgGrad and bpp	0.9608	0.9231	0.9561	0.9141	0.5929	0.3516

Synthetic images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
ImgGrad and PSNR	-0.9385	0.8807	-0.9385	0.8807	-0.9478	0.8983
ImgGrad and CR	-0.8866	0.7860	-0.8620	0.7431	-0.8476	0.7185
ImgGrad and bpp	0.9076	0.8238	0.8842	0.7818	0.8650	0.7483

Clipart images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
ImgGrad and PSNR	-0.7569	0.5881	-0.8436	0.7117	-0.6859	0.4705
ImgGrad and CR	-0.5645	0.3187	-0.5917	0.3502	-0.5181	0.2684
ImgGrad and bpp	0.7113	0.5060	0.7857	0.6174	0.5679	0.3225

Table 5.13 Image gradient correlation results for YC_bC_r images

Natural images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
ImgGrad and PSNR	-0.9405	0.8846	-0.6060	0.3672	-0.7990	0.6383
ImagGrad and CR	-0.8519	0.7258	-0.5196	0.2899	-0.5288	0.2796
ImagGrad and bpp	0.9435	0.8901	0.7521	0.5657	0.8070	0.6512

Synthetic images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
ImgGrad and PSNR	-0.9434	0.8900	-0.7218	0.5207	-0.5981	0.3577
ImagGrad and CR	-0.8544	0.7299	-0.7227	0.5223	-0.7944	0.6310
ImagGrad and bpp	0.8654	0.7489	0.8260	0.6822	0.8381	0.7023

Clipart images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
ImgGrad and PSNR	-0.8367	0.7001	-0.7649	0.5851	-0.5291	0.2799
ImagGrad and CR	-0.5989	0.3587	-0.5893	0.3473	-0.5656	0.3199
ImagGrad and bpp	0.7718	0.5957	0.8747	0.7651	0.8360	0.4045

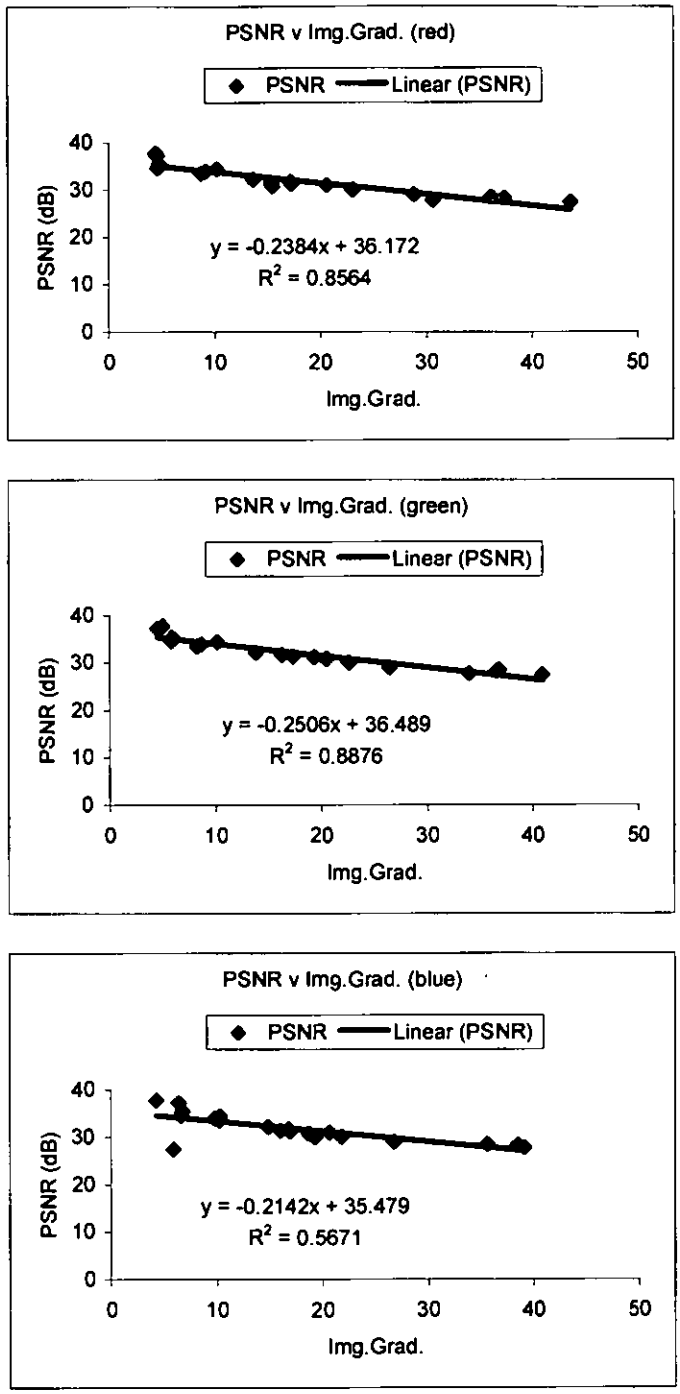


Figure 5.6 Scatter plots for image gradient and PSNR

The linear equation is used to establish the relationship between image gradient and PSNR. The equation can be used to predict the PSNR from the image gradient measure for a given image.

5.7. Spatial Frequency

An additional characteristic used to analyse the sample images is the spatial frequency (SF) in the spatial domain [Eskicioglu and Fisher, 1995]. SF is the mean difference between neighbouring pixels and it indicates the overall activity in an image. It is defined as:

$$SF = \sqrt{R^2 + C^2}$$

where $R = \sqrt{\frac{1}{MN}\sum_{j=1}^{M}\sum_{k=1}^{N}(X_{j,k} - X_{j,k-1})^2}$ and $C = \sqrt{\frac{1}{MN}\sum_{j=1}^{M}\sum_{k=1}^{N}(X_{j,k} - X_{j-1,k})^2}$.

5.7.1. Spatial Frequency Characteristics Results

The spatial frequency measures for a set of sixty different colour images were calculated in the RGB and YC_bC_r colour space. The following table shows a sample of the results of spatial frequency (SF) measures.

Table 5.14 Spatial frequency measure results

	RGB			YCbCr		
Image	SF (r)	SF (g)	SF (b)	Sf (Y)	SF (Cb)	SF (Cr)
bird	8.37	8.76	9.93	8.37	2.82	2.43
bulehills256	5.29	5.78	4.93	5.32	1.34	1.54
columns	17.63	17.15	18.78	17.15	2.72	3.32
snowwhite02	17.30	17.16	16.77	17.11	1.26	1.17
toystory02_256	27.86	25.95	22.89	25.46	5.81	5.78
toystory01_256	30.94	30.78	30.58	30.53	4.16	3.87
clipart09_256	52.01	66.37	69.23	54.85	31.50	26.61
clipart06_256	38.88	39.77	40.29	39.34	3.16	4.26
clipart10_256	51.54	47.97	47.23	48.88	1.38	2.57

To check if there is any relationship between SF and coding performance measures the r and r^2 values are computed and the results are presented in Table 5.15.

Table 5.15 SF correlation results for RGB images

Natural images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SF and PSNR	-0.9301	0.8650	-0.9414	0.8862	-0.8167	0.6670
SF and CR	-0.8544	0.7300	-0.8748	0.7653	-0.7473	0.5585
SF and bpp	0.9499	0.9023	0.9399	0.8835	0.6933	0.4807

Synthetic images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SF and PSNR	-0.7990	0.6385	-0.7891	0.6226	-0.7853	0.6167
SF and CR	-0.8453	0.7146	-0.8365	0.6997	-0.8351	0.6974
SF and bpp	0.9006	0.8111	0.8995	0.8091	0.8940	0.7992

Clipart images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SF and PSNR	-0.7656	0.5862	-0.7541	0.5687	-0.6850	0.4692
SF and CR	-0.8453	0.7146	-0.8365	0.6997	-0.8351	0.6974
SF and bpp	0.7773	0.6042	0.8075	0.6521	0.6690	0.4475

Table 5.16 SF correlation results for YC_bC_r images

Natural images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SF and PSNR	-0.9503	0.9030	-0.5777	0.3337	-0.7007	0.4910
SF and CR	-0.8801	0.7746	-0.5196	0.2699	-0.6920	0.4789
SF and bpp	0.9274	0.8600	0.7243	0.5247	0.7467	0.5576

Synthetic images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SF and PSNR	-0.7998	0.6397	-0.6572	0.4319	-0.5542	0.3071
SF and CR	-0.8402	0.7059	-0.7227	0.5223	-0.7192	0.5173
SF and bpp	0.8965	0.8037	0.7765	0.6029	0.8283	0.6828

Clipart images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SF and PSNR	-0.7712	0.5947	-0.7097	0.5036	-0.1412	0.0199
SF and CR	-0.6452	0.4162	-0.5893	0.3473	-0.3535	0.1250
SF and bpp	0.7779	0.6051	0.9002	0.8104	0.4470	0.1998

The analysis shows that the results obtained using the SF measure is similar to the ones obtained using image gradient. The results show there is a strong negative correlation between the SF and PSNR, and between SF and CR for natural and synthetic images. The results also show that there is a strong positive correlation between SF and bpp. The correlation for the blue colour component is not as strong as the red and green colour

components. There is a positive correlation between SF and all three coding performance measures for the red and green colour components for clipart images and a weaker correlation for the blue colour component.

Upon a closer examination of the definition of image gradient and spatial frequency, it could be seen that the definition for an image gradient and spatial frequency are basically the same. Hence, this could explain the similarities in the results.

5.8. Spectral Flatness Measure

The spectral flatness measure (SFM) of a digital image is defined as:

$$SFM(\Theta) = \frac{\left[\prod_{k=0}^{M-1}\prod_{l=0}^{N-1}|\theta(k,l)|^2\right]^{\frac{1}{MN}}}{\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|\theta(k,l)|^2} \tag{5.14}$$

where $\theta(k,l)$ refers to the Fourier coefficients [Sprljan et al, 2003].

The SFM provides an indication of the overall activity of a digital image. A digital image with a SFM of 1 indicates that all the pixels have the same value, except one. A lower *SFM* indicates that the energy of image is concentrated in fewer coefficients. Sprljan et al [2003] cited that if an image has a flat or near-flat spectrum then the quality of any prediction will be poor.

5.8.1. SFM Results

This section presents the results of the investigation into how the SFM measure affects the coding performance. Table 5.17 shows a sample of the results of SFM measures for a set of nine different colour images in the RGB and YC_bC_r colour space.

Table 5.17 SFM results in the RGB and YC_bC_r colour space

	RGB			YCbCr		
Image	SFM (r)	SFM (g)	SFM (b)	SFM (Y)	SFM (Cb)	SFM (Cr)
bird	0.000516	0.000498	0.000638	0.000429	0.013696	0.016693
bulehills256	0.000463	0.000209	0.000083	0.000217	0.000070	0.000137
columns	0.001432	0.001728	0.001542	0.001487	0.021050	0.027082
snowwhite02	0.001778	0.001962	0.002298	0.001917	0.002685	0.003848
toystory02_256	0.002812	0.002899	0.003912	0.002900	0.001845	0.003980
toystory01_256	0.014601	0.013187	0.013999	0.013620	0.011687	0.011979
clipart09_256	0.002897	0.005734	0.003989	0.003281	0.025576	0.001348
clipart06_256	0.003676	0.003829	0.003961	0.003752	0.108185	0.066778
clipart10_256	0.009834	0.009319	0.009202	0.009464	0.015427	0.015427

Like the previous sections, the r and r^2 values are computed to check if there is any relationship between SFM and coding performance measures and a sample of the results is presented in Table 5.18.

Table 5.18 SFM correlation results in the RGB colour space

Natural imagee:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SFM and PSNR	-0.5936	0.3523	-0.7300	0.5330	-0.7007	0.4910
SFM and CR	-0.4484	0.2010	-0.5929	0.3515	-0.6578	0.4327
SFM and bpp	0.5040	0.2540	0.6160	0.3795	0.6478	0.4196

Synthetic images:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SFM and PSNR	-0.6887	0.4743	-0.6273	0.3935	-0.4392	0.1929
SFM and CR	-0.6886	0.4742	-0.6790	0.4611	-0.4949	0.2449
SFM and bpp	0.7835	0.6138	0.7528	0.5667	0.5547	0.3077

Clipart imagee:

	r (red)	r^2(red)	r (green)	r^2(green)	r (blue)	r^2(blue)
SFM and PSNR	-0.6989	0.4885	-0.6934	0.4808	-0.6160	0.3795
SFM and CR	-0.3956	0.1565	-0.4440	0.1972	-0.3736	0.1396
SFM and bpp	0.5801	0.3365	0.6296	0.3964	0.4694	0.2204

Table 5.19 SFM correlation results in the YC_bC_r colour space

Natural images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SFM and PSNR	-0.7149	0.5111	-0.4141	0.1715	-0.4727	0.2235
SFM and CR	-0.5774	0.3334	-0.3308	0.1095	-0.3929	0.1544
SFM and bpp	0.6213	0.3860	0.4349	0.1892	0.6680	0.4462

Synthetic images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SFM and PSNR	-0.6517	0.4246	-0.5344	0.2856	-0.5500	0.3025
SFM and CR	-0.6959	0.4842	-0.5110	0.2611	-0.4498	0.2023
SFM and bpp	0.7871	0.6196	0.6093	0.3712	0.6093	0.3712

Clipart images:

	r (Y)	r^2(Y)	r (Cb)	r^2(Cb)	r (Cr)	r^2(Cr)
SFM and PSNR	-0.7019	0.4927	-0.2205	0.0486	-0.2739	0.0750
SFM and CR	-0.4158	0.1729	-0.0299	0.0009	-0.0822	0.0067
SFM and bpp	0.6004	0.3605	-0.0063	0.0000	0.0346	0.0012

Comparing the results in Table 5.18 with the results in Table 5.19 shows that there is a negative correlation between the SFM and PSNR, and between SFM and CR for natural and synthetic images. The results also show that there is a positive correlation between SFM and bpp. It is also observed that there is a weaker correlation for the blue colour component compared to the red and green colour component.

5.9. Summary

The results from section 5.4.3 show that the best combination of PSNR and CR is produced by the '`bior4.4`' (9/7 tap) wavelet filter. This result is wholly consistent with the selection of the Daubechies 9/7 tap wavelet as the basis of the JPEG2000 standard. Furthermore, it is also found that the `haar` wavelet filter outperforms the other wavelets for clipart images.

Also in this chapter a variety of grey-level image histogram statistics is used to analyse sixty colour images. The aim is to determine what causes the variation in coding performance measures. It is observed that most of the grey-level image histogram statistics, that is, mean, standard deviation, variance, skewness, and kurtosis do not have any significant connection to the coding performance measures. The results have shown that image gradient and spatial frequency has a strong correlation with all three coding

performance measures. The strong correlation between image gradient and all three coding performance measures support the findings of Saha and Vemuri [2000b] for grey-scaled images. However, the equation used to describe the relationship between image gradient and PSNR is a linear equation instead of a logarithmic equation used by Saha and Vemuri [2000b].

Although the results in this chapter show that image gradient and spatial frequency has a strong correlation with all three coding performance measures, this research is cautious not to take this as evidence of causation and the result remains as a statistical association. It is therefore proposed that there should be a more through study conducted to search for the cause of this correlation. However, this research is concerned with image compression and thus will not pursue further the cause of this correlation. The basic findings are however relevant to this research and the results from this chapter inform the choice of wavelet and other aspects of the development of three wavelet-based compression systems reported in chapter 6 and 7.

CHAPTER 6

Approaches to Compressing Grey-Scale and Colour Images

6.1. Introduction

This chapter presents a method for identifying the frequency characteristics of a colour image and two methods of compressing grey-scale and colour images using a wavelet-based segmentation approach.

6.2. Analysis of an Image Using a Block-based FFT Method

This section is concern with a systematic study conducted to identify a colour image according to its image activity. A method using a block-based Fast-Fourier transform (FFT) is used to determine the image activity. The results confirm that an image with a low-level of activity is easier to code compared to an image with a high level of activity.

6.2.1. Histogram Statistical Features

The work is discussed in detail in chapter 5 and explores how various image characteristics influence the coding performance measures using grey-scale image histogram features to analyse colour images. The histogram features are mean, standard deviation, variance, energy, skew, kurtosis, and entropy. The results showed that the images from each category defined in this research had similar characteristics. There were no dominant statistical features that could be used to distinguish between them. The results also showed that these statistical features did not correlate well with coding performance criteria like PSNR (which was used as an indicator of image quality) and bpp (which was used as an indicator of compression performance). As these global statistical features could not explain the coding performance, other image characteristics were needed and it was decided to explore in the frequency domain based on local image properties.

6.2.2. Image Characteristics in the Frequency Domain

A typical RGB colour image can be considered to comprise three independent grey-scale images, where each image corresponds to a different colour. Typically, each of these grey-scale images will consist of pixels that are correlated and therefore contain redundant information. Significant compression can be achieved by exploiting these redundancies,

the basis of almost all compression methods. A detailed examination of the statistical characteristics of the image pixels may provide useful information pertaining to the possible level of compression of the image concerned.

A Fourier analysis of an image can help appreciate its statistical characteristics. However, a global Fourier analysis of an entire image provides very little useful information due in part to the large averaging effect and because the statistics of a typical image are not stationary. Kingsbury cited a better approach that is to split the grey-scale image into blocks and calculate the Fourier log power spectrum of each block [Kingsbury, 2003].

6.2.3. Method and Results

In this study, Kingsbury's [2003] approach is extended to RGB images. The image concerned is first split into the R, G, and B components. The three individual components are then divided into 16x16 blocks. The 2D-FFT is then applied to the blocks and the Fourier log power spectrum computed. The 2D-FFT is defined by:

$$F(u,v)=\frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}F(j,k)\exp\left\{\frac{-2\pi i}{N}(uj+vk)\right\}$$

where $i=\sqrt{-1}$, the indices (u,v) are the spatial frequencies of the transformation and the indices (j,k) are the pixel positions. Figure 6.1 is used to demonstrate this approach.

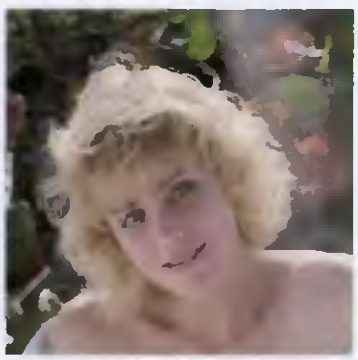
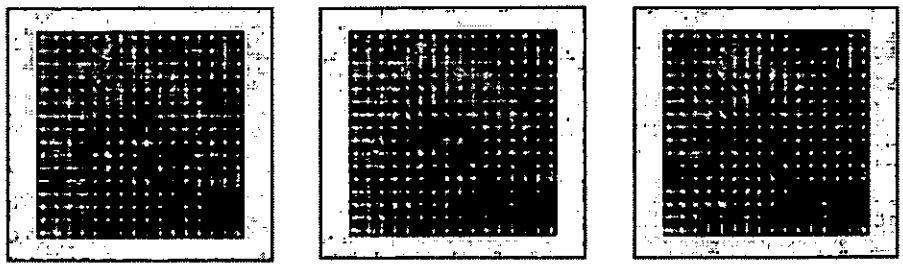


Figure 6.1 Original image [source: Eastman Kodak Co.]



Fourier Log Power Spectra of R component

Fourier Log Power Spectra of G component

Fourier Log Power Spectra of B component

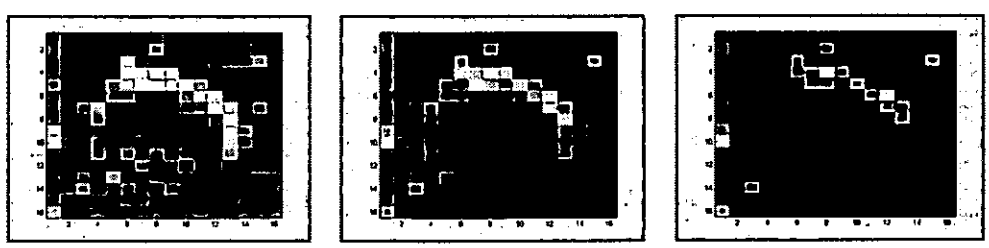
Figure 6.2 Fourier log power spectra images

The above images indicate a wide variation in spectra. The bright center in the blocks, indicate a low frequency component. The blocks that contain the hair produce a spread of energy at all frequencies. Spectra with lines normal to the edges indicate the presence of dominant edges, for example the transition between shoulder and background.

The spectra images contain no useful statistical information that could help define the coding performance. To analyze the image more effectively, a more quantifiable measure is needed. To provide this, the average magnitude (AM) of each block is computed. The array of AM can then be displayed as an image. The average magnitude is defined as:

$$\text{Average Magnitude} = \sum_{j,k} |F(j,k)| / N$$

where $|F(j,k)|$ is the matrix containing the amplitudes of the spectrum and N is the number of frequency components [Augusteijn et al, 1995; Dulyakarn et al, 2000]. The corresponding AM images of the Fourier log power spectra images shown in Figure 6.3, is shown here.



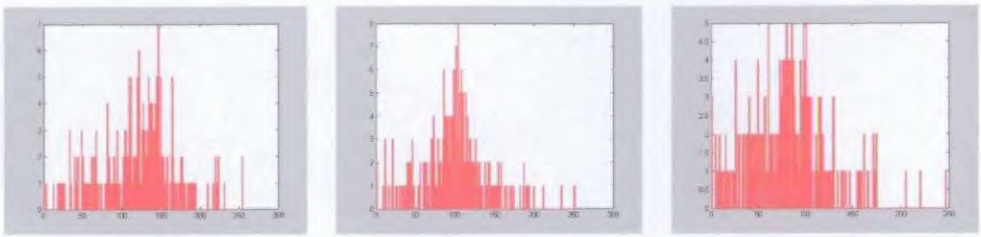
Average Magnitude of R component

Average Magnitude of G component

Average Magnitude of B component

Figure 6.3 Average magnitude images

Next, the 2D-array of AM is converted into a 1D-array and a histogram is produced.



AM Histogram of R component | AM Histogram of G component | AM Histogram of B component

Figure 6.4 AM histograms

To determine the level of activity of the image, the histogram is divided arbitrarily into three bands. Each band consists of the frequency of number of average magnitude values.

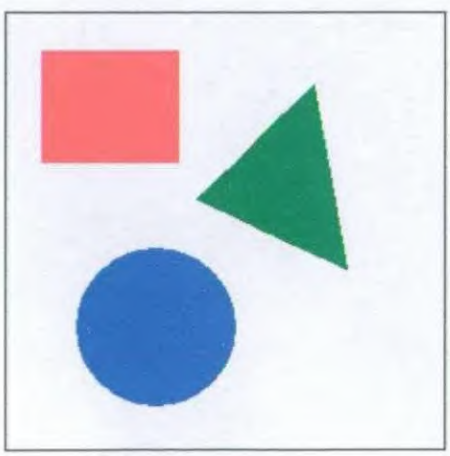
Table 6.1 Image analysis using block processing FFT technique

Low band: 148	Moderate band: 94	High band: 14
Low band: 149	Moderate band: 93	High band: 14
Low band: 147	Moderate band: 87	High band: 22
Ave. Low: 148.00 (58%)	Ave. Moderate: 91.33 (36%)	Ave. High: 16.66 (6%)

From the results, the image can be classified as having low to moderate activity.

Six test images (Figure 6.5) were analysed using the above method and the results are presented in Table 6.2.

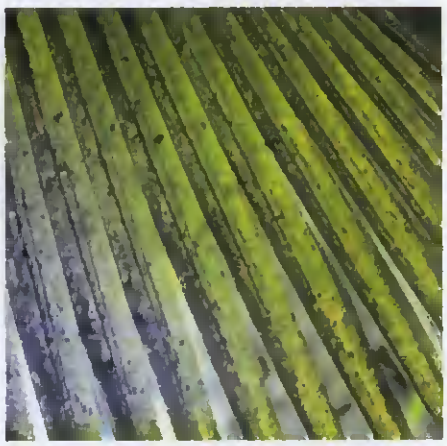
bird



shapes01

peppers

marcie



palmblades

mandrill

Figure 6.5 Colour test images

Table 6.2 Analysis of images

Image	Low band	Moderate band	High band
bird	212.33 (83%)	35.66 (14%)	8.00 (3%)
shapes01	216.66 (85%)	15.00 (6%)	24.33 (9%)
marcie	148.00 (58%)	91.33 (36%)	16.66 (6%)
peppers	71.66 (28%)	136.66 (53%)	47.66 (19%)
mandrill	5.33 (2%)	135.33 (53%)	115.33 (45%)
palmblades	73.66 (29%)	73.66 (29%)	108.66 (42%)

The results show that images like 'mandrill' that contain a lot of detail, comprise a large number of high (45%) and moderate (53%) average magnitudes. It is concluded that high average magnitudes are indicators of a high level of image activity. Images like 'bird' that

contain a large smooth region have a large number of small average magnitudes (83%), which indicates a low level of activity.

These images were then compressed using different wavelets. The results for four wavelets are presented here.

Table 6.3 Compression results

Image	haar			db2			bior4.4			bior6.8		
	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp
bird	36.66	45.18	0.18	36.98	44.65	0.18	37.23	41.57	0.19	37.26	32.67	0.25
shapes01	41.53	35.18	0.23	35.96	26.87	0.30	34.36	27.33	0.29	34.37	25.66	0.31
marcie	31.08	19.16	0.42	31.77	19.78	0.40	32.03	17.77	0.45	32.18	13.68	0.59
peppers	30.28	13.40	0.60	30.90	14.57	0.55	31.29	12.98	0.62	31.32	9.92	0.81
mandrill	25.87	7.26	1.10	25.97	7.26	1.10	25.97	6.77	1.18	26.02	5.85	1.37
palmblades	26.67	6.72	1.19	26.96	7.25	1.10	27.27	7.21	1.11	27.40	6.06	1.32

The results confirm that images that have a high level of activity are harder to code. High levels of activity mean that there is more information compared to a low-level activity image, which has low information content.

6.3. A Dual Wavelet Compression Scheme for Still Colour Images

Conventional wavelet-based compression schemes first convert a colour RGB image into luminance/chrominance format before the wavelet transform is carried out. The luminance/chrominance image is decomposed with a single wavelet filter and at a single decomposition level. The work described in this section demonstrates that this does not necessarily achieve optimum coding performance. A new scheme is proposed that uses one wavelet to compress the luminance image and another to compress the chrominance images. The results also show that coding performance can be improved with a single wavelet by processing the luminance and chrominance images at different levels of decomposition.

To compress colour images a wavelet-based encoder would comprise of a colour transform, a wavelet transform, a quantizer, and an entropy coder.

Figure 6.6 Typical wavelet-based encoder

Typically all colour images are in the RGB colour space. The RGB image has to be transformed into the YC_bC_r colour space before being decomposed into wavelet coefficients, which are then quantised. The entropy coder produces an output bit stream and then encodes these wavelet coefficients.

This typical approach applies the same wavelet filter to decompose the Y, C_b, and C_r components without considering that the human visual system (HSV) is relatively insensitive to distortion in the chrominance components especially at high frequencies. The C_bC_r chrominance components have a minor role in the perception of edges and fine details [Taubman et al, 2002, pp.421]. Lervick and Ramstad [1995] show that the C_bC_r components generally have a smaller amplitude dynamic range than the Y component. Lervick and Ramstad [1995] also show that the amplitudes of the C_bC_r are also smaller than the Y component. In addition, the human vision system (HVS) is less sensitive to intensity changes than for spatial variations in the Y component. This property can be exploited to help reduce the encoded output.

The proposed wavelet-based image coder is built on the two-dimensional wavelet transform discussed in section 3.3.5. It supports tri-component images and also supports a variety of wavelet families supported by Matlab. The main components are colour transform, wavelet transform, threshold, quantizer and entropy encoder.

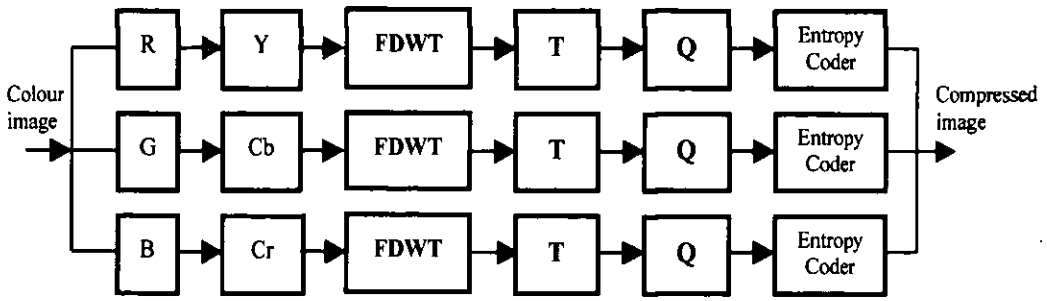


Figure 6.7 Structure of the wavelet-based tri-component coder

The equation used to transform the RGB image into YC_bC_r colour space is defined in section 3.9.2. A two-dimensional discrete wavelet transform (DWT) wavelet analysis function is used to decompose the Y, C_b, and C_r components separately. Since the resultant wavelet coefficients after DWT are close to zero these coefficients can be set to zero by using a method called thresholding.

After transformation and thresholding, all the coefficients are quantized. The coefficients are floating-point values and these are scaled and quantified. Each coefficient is represented by *n* bits, which is some value between 8 and 16 bits. The scaling works by computing the maximum and minimum of the coefficients and maps the coefficients to the quantization steps. The number of steps is determined from the value *n*. The quantization process maps the coefficients into integer values, which is then fed into the Huffman entropy coder.

A Huffman entropy-coding scheme is used to reduce the storage size by reducing the number of bits required to code the coefficients. The coefficient matrix is converted into a long single sequence of integer values, and these values are encoded as symbols. The length of the code words is first computed, and then the Huffman code words are determined. Finally a Huffman tree is constructed and is used for the compression.

The decoding scheme is basically the reverse of the encoding process (See Figure 6.8). The compressed image is first decompressed using the Huffman decoding scheme to obtain the coefficient matrix. The coefficients are then re-scaled in the dequantization stage and the inverse DWT (IDWT) is carried out to reconstruct the image. Following the IDWT, an

inverse mapping is performed from the Y, C_b, and C_r colour space to the RGB space using the following equation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & -0.71414 \\ 1.0 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

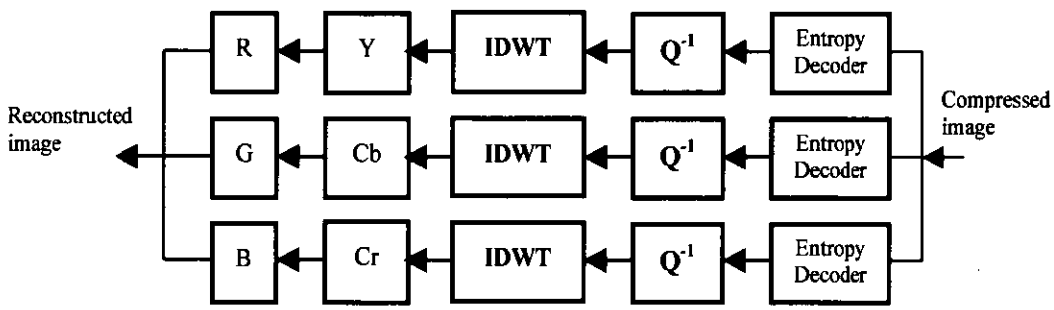


Figure 6.8 Decoding structure

With this scheme a different wavelet filter can be used to decompose the luminance and chrominance components separately. This is consistent with the work of Lervik and Ramstad [1995], which cited that the statistics of the C_bC_r components are equal and they use the same subband filter bank for both the C_bC_r components. However, in the work reported here, it is shown that there may be some benefits in using different wavelet filter for each of the C_bC_r components, and different levels of decomposition.

The six images used in this study were classified as low, moderate, and high frequency, using a block-based FFT analysis method described in section 6.2.3. The frequency spectrum was calculated for each of the Y, C_b and C_r image planes separately and then aggregated to produce the final figure. Table 6.4 presents the results. Earlier work on grey scale images had indicated that significant improvements in PSNR and/or the compression ratio (CR) were possible through the selection of different wavelets, based on the frequency characteristics of an image (see Table 6.3).

Table 6.4 Image Classification

	Image	Low band	Moderate band	High band
Low Fraquency	bird	235 (92%)	17.66 (7%)	3.33 (1%)
	flower256	246 (96%)	2.66 (1%)	7.33 (3%)
Moderate Frequency	marcie	129 (50%)	113.66 (45%)	13.33 (5%)
	peppers	83.33 (32%)	132 (52%)	40.66 (16%)
High Frequency	barboon	50.667 (20%)	139 (54%)	66.33 (26%)
	palmblades	10 (4%)	134.66 (53%)	111.33 (43%)

To provide a basis for comparison, the principal wavelet types supported by the Matlab package were applied to the six images, using level 3 decomposition and reconstruction filters. The results are shown in Table 6.5. It can be seen that the best CR is produced by the `haar` wavelet for low frequency images and `db2` for the moderate and high frequency images. The best PSNRs are produced by the `bior4.4` and `sym4`, but with no particular pattern being evident. Overall, as in section 5.4.3 the best combination of PSNR and CR is produced by the `bior4.4` (9/7 tap) wavelet.

Table 6.5 Coding performance using a single wavelet

Image	haar			db2			bior4.4			sym4			coif4		
	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp
bird	36.66	45.18	0.18	36.98	44.65	0.18	37.23	41.57	0.19	37.29	40.90	0.20	37.01	28.09	0.29
flower256	36.70	58.78	0.14	37.32	56.86	0.14	37.22	47.25	0.17	37.18	50.92	0.16	37.12	31.98	0.25
marcie	31.08	19.16	0.42	31.77	19.78	0.40	32.03	17.77	0.45	32.01	17.98	0.45	32.03	11.46	0.70
peppers	30.28	13.40	0.60	30.90	14.57	0.55	31.29	12.98	0.62	31.23	13.06	0.61	31.23	8.18	0.98
mandrill	25.87	7.28	1.10	25.97	7.26	1.10	25.97	6.77	1.18	26.00	6.89	1.16	26.01	5.24	1.53
palmblades	26.87	6.72	1.19	26.96	7.25	1.10	27.27	7.21	1.11	27.19	7.10	1.13	27.25	5.34	1.50

The next stage was to combine the wavelets producing the best PSNRs with those producing the best CRs, that is the `bior4.4` with the `haar` and `db2`. The luminance (Y) image plane contains the majority of the image details and it was therefore decided that, for this study, the wavelet producing the best PSNR should be used for this element of the compression (that is the `bior4.4`). The `haar` and `db2` wavelets were used to compress both of the chrominance planes, C_b and C_r. The results obtained are shown in Table 6.6.

Table 6.6 Coding performance using dual wavelets

	bior4.4(3), haar			bior4.4(3), db2		
Image	PSNR	CR	bpp	PSNR	CR	bpp
bird	36.92	46.36	0.17	37.21	44.38	0.18
flower256	37.22	54.69	0.15	37.24	53.75	0.15
marcie	31.72	19.34	0.41	31.93	19.04	0.42
peppers	30.57	13.45	0.60	31.08	13.97	0.57
mandrill	25.85	6.88	1.16	25.94	6.92	1.16
palmblades	26.57	7.04	1.14	26.71	7.27	1.10

The results in Table 6.6 show that both combinations show an improvement in the compression ratio with only a small loss in PSNR, compared to the `bior4.4`. The exception to this is palmblades (`bior4.4` and `haar`). The best improvement is for the `bior4.4`(3), `haar`(3) for the flowers256 image, which shows an increase in CR of 15.7% for no loss of PSNR. Overall, the `bior4.4`(3), `db2`(3) produces an increase in CR of 8.10% for a loss of only 0.56% in PSNR. The `bior4.4`(3), `haar`(3) produces overall increase in CR of 9.62% but the PSNR is reduced by 1.13%.

In a further experiment, it was found that a combination of `bior4.4`, `db2`, `db3` can marginally improve the CR for mandrill. The results are shown in Table 6.7.

Table 6.7 Triple wavelet coding performance

	bior4.4(3), db2(3), db3(3)		
Image	PSNR	CR	bpp
bird	37.18	45.01	0.18
flower256	37.23	52.81	0.15
marcie	31.94	18.78	0.43
peppers	31.18	13.84	0.58
mandrill	25.94	6.93	1.15
palmblades	26.70	7.25	1.10

This could open up a possibility of a triple wavelet compression system. As the results in Table 6.7 show only a marginal improvement in CR for a particular image for a significant increase in complexity, this approach will not be considered any further.

Finally, a series of results were taken for a single wavelet (`bior4.4`) but with different levels of decomposition applied to the chrominance components (C_bC_r). The `bior4.4`(3,7) produces a large overall gain in CR (22.45%) but at an large fall in PSNR

(3.48%). The overall figure for `bior4.4`,(3,5) produces an overall increase in CR of 23.22% for a loss in PSNR of 0.77%. The best individual performance is again for flowers256, where an increase of 32.5% in CR has been produced for a loss of only 0.34% in PSNR.

Table 6.8 Coding performance using different levels of decomposition

	bio4.4 (3,1)			bior4.4 (3,3)			bior4.4 (3,S)			bior4.4 (3,7)		
Image	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp	PSNR	CR	bpp
bird	28.13	32.37	0.2S	37.23	41.S7	0.19	37.08	S3.92	0.1S	36.99	56.03	0.14
flower2S6	29.8S	17.S8	0.46	37.22	47.25	0.17	37.20	62.59	0.13	37.09	65.23	0.12
marcie	30.S9	7.98	1.00	32.03	17.77	0.4S	31.86	19.51	0.41	31.74	19.38	0.41
peppers	31.SS	6.06	1.32	31.29	12.98	0.62	30.84	13.92	0.58	29.58	15.29	0.52
mandrill	26.07	4.75	1.68	2S.97	6.77	1.18	25.71	7.16	1.12	25.04	7.67	1.04
palmblades	27.80	S.41	1.48	27.27	7.21	1.11	26.8S	7.4S	1.07	23.92	8.61	0.93

The above results, although confined to a small sample of images and range of wavelets, clearly demonstrate the potential gains that may be achieved by the application of either multiple wavelets or different levels of decomposition of the same wavelet. The image samples were chosen to give a good statistical spread of characteristics which gives confidence of the general applicability of this technique. The overall results for `bior4.4` (3,5) are particularly impressive.

6.4. A Segmentation-based Wavelet Compression Scheme for Grey-Scale Images

The advent of picture messaging on mobile telephones, Personal Digital Assistant (PDA) technology and other wireless based image services has shifted the balance of priority for image compression schemes. Historically, the output medium for decompressed images was a high resolution computer monitor that demanded a good quality image. Image quality can be broadly interpreted to mean a good Peak Signal to Noise Ratio (PSNR). The capabilities of the display screen on the majority of mobile devices combined with the cost of transmission bandwidth has shifted the emphasis away from the PSNR towards the level of compression, frequently expressed as a direct compression ratio or via a bits per pixel (bpp) value.

In recent years wavelet-based compression has become popular because such compression schemes allow high compression ratios while maintaining good image quality. The popularity and success of wavelets has prompted its inclusion in the JPEG2000 standard. A

variation on conventional wavelet-based image compression schemes is presented here. The proposed technique exploits the variable frequency characteristics of an image by applying different wavelet filters to the low and high frequency elements. It is shown that this can offer an improvement in the compression ratio for certain types of image, with little overall loss in PSNR.

The conventional lossy wavelet compression approach uses the same wavelet filter to compress the whole image. This section proposes a compression scheme based on the segmentation of an image into its non-smooth and smooth segments. A different wavelet filter is then applied to these different elements of the image.

An edge-detector algorithm was used to segment the grey-scale image into smooth and non-smooth segment images. A smooth segment image is defined as an image with smooth varying, continuous surfaces. In contrast, a non-smooth image consists of distinctive sharp edges, for example clipart. Since edges are usually sharp changes, an edge-detector can be used to identify fast gradient changes. The gradient change of an image can be derived by computing its first derivative by finding the numerical approximation of the difference in each pixel. The gradient change can be computed by:

$$\sqrt{[I(r,c)-I(r-1,c-1)]^2+[I(r,c-1)-I(r-1,c)]^2}$$

$I(r,c)$ denotes an image pixel at row (r) and column (c).
Typically, an approximate magnitude is computed using:

$$|I(r,c)-I(r-1,c-1)|+|I(r,c-1)-I(r-1,c)|$$

This form of the equation is generally preferred because it is much faster to compute [Umbaugh, 1998, pp.64].

Edge-detectors work on the basis that edge information in an image can be located by examining the relationship of a particular pixel and the surrounding pixels. If there is a wide variation of grey levels surrounding a pixel then an edge is present. On the other hand, if the grey levels are similar then there is no edge present at that point.

The following is a modified version of an edge detection algorithm scheme first presented by Liu (2004). A Roberts operator, chosen because it works best with grey-scale images [Umbaugh, 1998, pp.63], is used to extract the edges. The Roberts operator mask is defined as:

$$\text{row mask} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{column mask} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

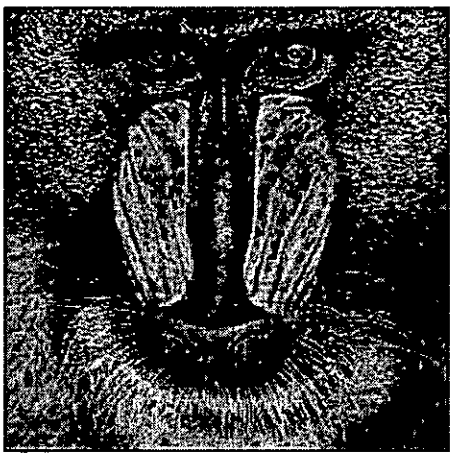
The Roberts operator convolution masks are convolved with the grey-scale image to produce an edge-detected image, which is basically a binary image. The resultant edge-detected image is then divided into blocks; values of 8 x 8, 16 x 16 and 32 x 32 were used in this study. The number of black pixels in each block is counted. If the block contains more than 10 black pixels, then it is considered to be a non-smooth segment. However, if the block contains less than 10 black pixels, then that region is said to be a smooth segment. Figure 6.9 provides an example of its operation.

Figure 6.9 Segmented images

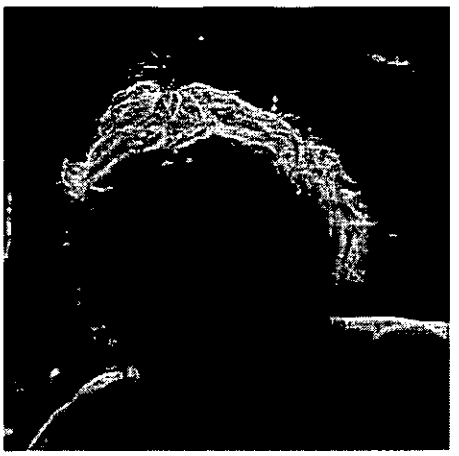
The smooth and non-smooth segment images were then compressed separately using a different wavelet filter for each image. The following is a comparison of the coding performance using a single wavelet compression scheme and the dual wavelet method presented above.

6.4.1. Grey-scale Image Segmentation-based Results

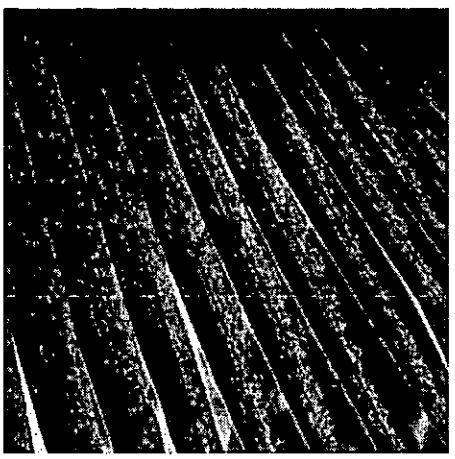
The segmentation scheme described above was implemented using Matlab, using two of the wavelets available in that package. The following six grey-scale images were used.



gmandrill



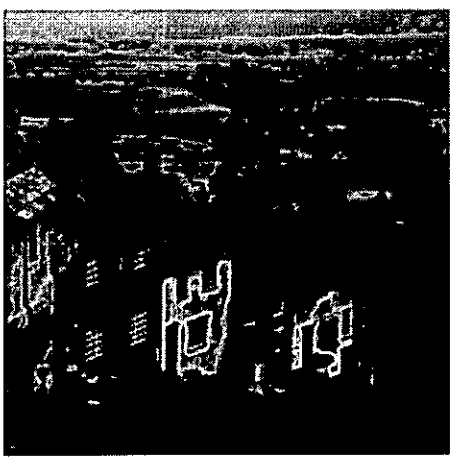
gmarcie



gpalmblades

gbarbara

gboat



ggoldhill

Figure 6.10 Grey-scale test images

The first set of results (Figure 6.11) gives a comparison of the output from the encoder for a single wavelet and the dual wavelet scheme. The number of bytes coming from the encoder has been used to provide an absolute measure of performance rather than the compression ratio, as is more normal. The reason for this is that the dual wavelet method produces two image outputs that must be added to produce the total data size. The `bior4.4` wavelet was used for the single wavelet method at a decomposition level of 3. For the dual wavelet method, the `bior4.4` was again used at level 3 for the smooth image and a `haar` wavelet at a decomposition level of 4, for the non-smooth segment image.

Image	Encoder output (single wavelet)	Encoder output (dual wavelet)
gmandrill	19125	14704
gmarcie	6849	5850
gpalmblades	15381	13686
gbarbara	11504	9286
gboat	8162	7399
ggoldhill	6889	5970

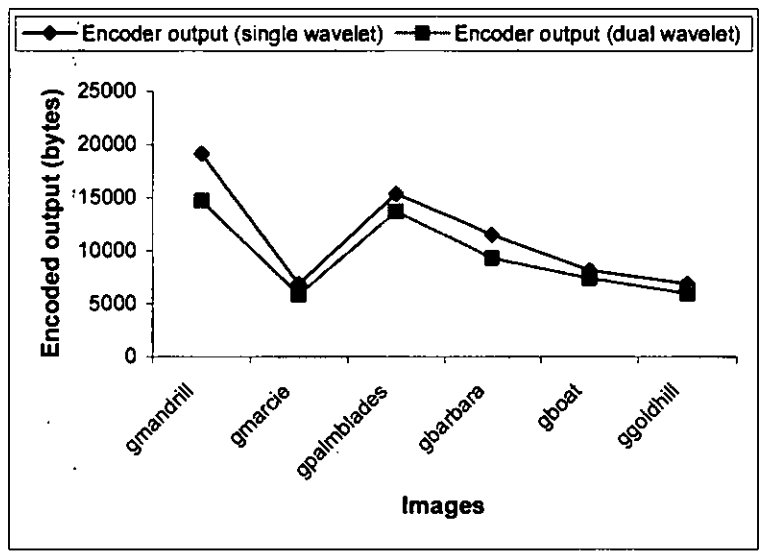


Figure 6.11 Comparison of resultant compressed file sizes for the single and dual wavelet methods.

From Figure 6.11 it is apparent that there is some improvement in compression as the total number of bytes coming from the encoder is smaller for the dual wavelet method. However, a direct comparison cannot be made as the effects of the application of the dual wavelet reduce the overall PSNR by approximately 2dB for each of the images.

Image	PSNR (single wavelet)	PSNR (dual wavelet)
gmandrill	29.20	27.14
gmarcia	32.93	30.37
gpalmbladas	29.31	27.77
gbarbara	31.29	28.55
gboat	31.78	30.17
ggoldhill	31.52	29.53

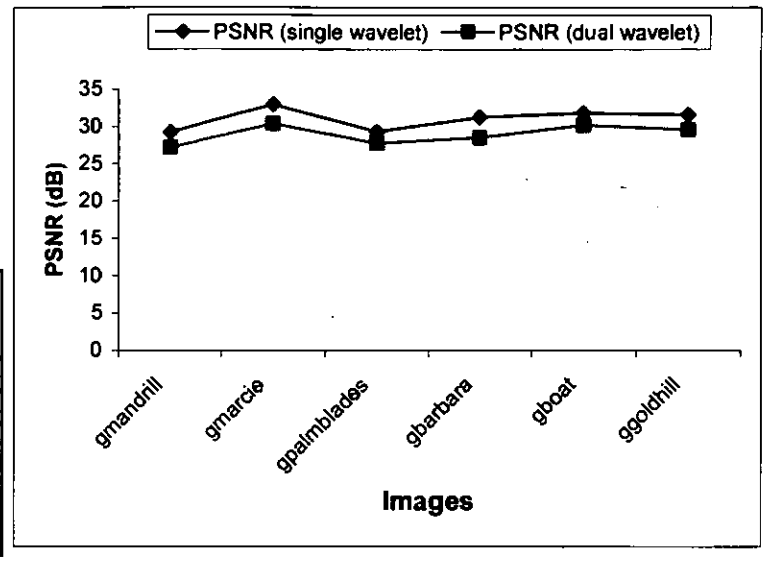


Figure 6.12 Reduction in PSNR caused by the dual wavelet method.

In order to provide a direct comparison, the PSNR for the single wavelet method was set to the same level as that achieved by the dual method, and the number of bytes produced by the encoder was recomputed. The results are shown in Figure 6.13.

Image	Encoder output (single wavelat)	Encoder output (dual wavelet)
gmandrill	15574	14704
gmarcie	4993	5850
gpalmblades	12874	13686
gbarbara	8674	9286
gboat	6590	7399
ggoldhill	4969	5970

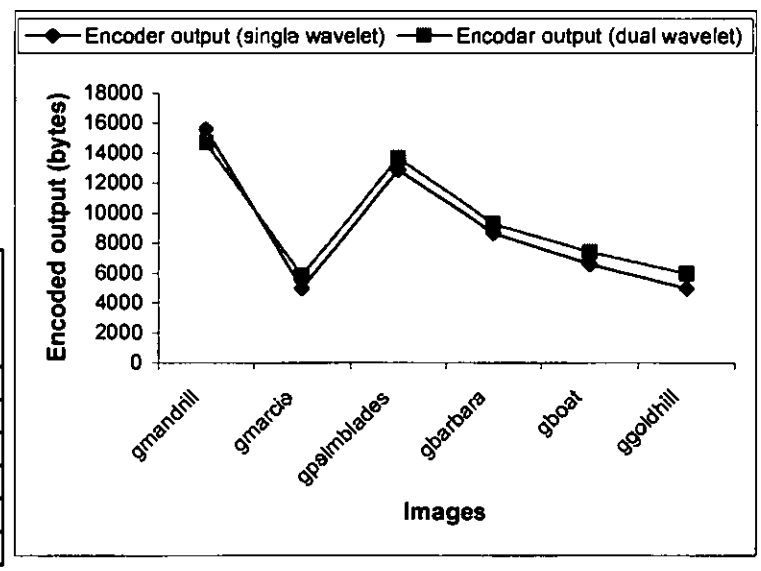


Figure 6.13 Comparison of the single and dual wavelet methods for fixed PSNR levels.

Initial results indicate that some improvement in compression can be achieved through the use of a combination of wavelets with little loss in the quality of the reconstructed image. This could prove to be a very significant factor for mobile devices, where bandwidth costs can be high and small reductions in PSNR would have little, if any, impact.

This technique can be extended to colour images where it is anticipated that more significant improvements may be possible. It is hoped at this stage further work can be carried out to classify regions of an image based on their frequency content, and then to match a small family of wavelets to these regions in order to optimise overall performance. One possible outcome from this study is to develop a simple expert system to perform the initial analysis on an image and then to assign the most appropriate combination of wavelets for its compression.

6.5. A Segmentation-based Wavelet Compression Scheme for Still Colour Images

This section describes an extension of the above scheme to still colour images. There are two major parts to the proposed scheme. Part 1 consists of a segmentation scheme described in section 6.3.1 to segment the luminance (Y) image into smooth and non-smooth segments.

Part 2 consists of the wavelet-based encoder and decoder described in section 6.3. A 2D DWT is used to decompose the segmented Y, Cb, and Cr components. The resultant wavelet coefficients are set to zero using thresholding and then quantized. Finally the storage size is reduced by a Huffman entropy-coding scheme. The process to recover the image is identical to the process is described in section 6.3.

6.5.1. Colour Image Segmentation-based Method Results

A block-based FFT method was used to classify the six images used in this study. The frequency spectrum was calculated for each of the Y, C_b and C_r image planes separately and then aggregated to produce the final value. Table 6.9 presents the results. Earlier work on grey scale images had indicated that significant improvements in PSNR and/or the compression ratio (CR) were possible through the selection of different wavelets, based on the frequency characteristics of an image [Yap and Comley, 2004].

Table 6.9 Colour Image Classification

	Image	Low band	Moderate band	High band
Low Frequency	bird	235 (92%)	17.66 (7%)	3.33 (1%)
	flower256	246 (96%)	2.66 (1%)	7.33 (3%)
Moderate Frequency	marcie	129 (50%)	113.66 (45%)	13.33 (5%)
	peppers	83.33 (32%)	132 (52%)	40.66 (16%)
High Frequency	mandrill	50.667 (20%)	139 (54%)	66.33 (26%)
	palmblades	10 (4%)	134.66 (53%)	111.33 (43%)

The principal wavelet types supported by the Matlab package were applied to the six images, using three levels of decomposition. The results are shown in Table 6.10 and they are used as a basis for comparison. The results show that the best encoded output is produced by the `haar` wavelet for low frequency images and `db2` for the moderate and high frequency images. The `bior4.4` wavelet filter yield the best PSNRs. Overall, the `bior4.4` (9/7 tap) wavelet filter yields the best combination of PSNR and CR. Again, this is wholly consistent with the selection of the 9/7 tap wavelet as the basis of the JPEG2000 standard.

Table 6.10 Coding performance using a single wavelet

	haar		db2		bior4.4		sym4		colf4	
Image	Encoded output	PSNR	Encoded output	PSNR	Encoded output	PSNR	Encoded output	PSNR	Encoded output	PSNR
bird	4352	36.66	4403	36.98	4730	37.23	4807	37.29	7000	37.01
flower256	3345	36.70	3458	37.32	4161	37.22	3861	37.18	6147	37.12
marcie	10263	31.08	9940	31.77	11066	32.03	10933	32.01	17163	32.03
peppers	14672	30.28	13490	30.90	15149	31.29	15057	31.23	24096	31.23
mandrill	27093	25.87	27093	25.97	29063	25.97	28554	26.00	37554	26.01
palmblades	29239	26.67	27106	26.96	27257	27.27	27699	27.19	36824	27.25

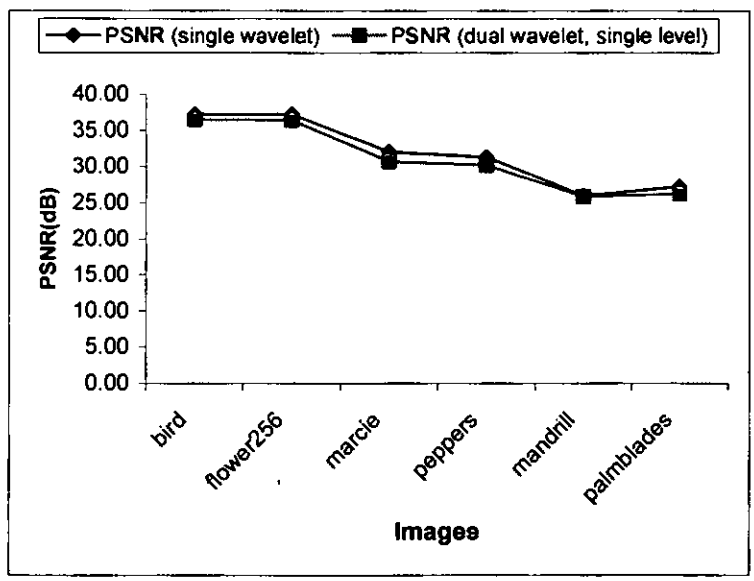
The next stage was to segment the Y image using the method described in section 6.4. The segmented Y image is compressed using the wavelet filters producing the best PSNRs with those producing the best CRs, that is, the `bior4.4` with the `haar` and `db2`. The smooth segments of the Y image, that contains the smooth/continuous features, are compressed with the `bior4.4` wavelet filter. The non-smooth segments, that contain most of the sharp features, are compressed with the `haar` or `db2` wavelet filter. The chrominance images, Cb and Cr are compressed with the `bior4.4` wavelet filter at level 3. The results obtained are shown in Table 6.11.

Table 6.11 Coding performance using segment-based approach, single level decomposition

	bior4.4(3), haar(3), bior4.4(3)		bior4.4(3), db2(3), bior4.4(3)	
	Encoded ouput		Encoded ouput	
Image	Total	PSNR	Total	PSNR
bird	4955	36.79	4652	36.49
flower256	4321	36.83	3874	36.39
marcie	11246	31.90	10087	30.67
peppers	15338	31.20	14283	30.18
mandrill	28786	25.93	29275	25.85
palmblades	28958	27.14	26407	26.25

Note: First two wavelets are used to decompose the segmented Y image at level 3 and level 4.
The third wavelet is used to decompose the chrominance components at level 3.

Figure 6.14 gives a comparison of the results in Tables 6.10 and 6.11 and shows that a there is a reduction in the encoded output with some loss in PSNR.



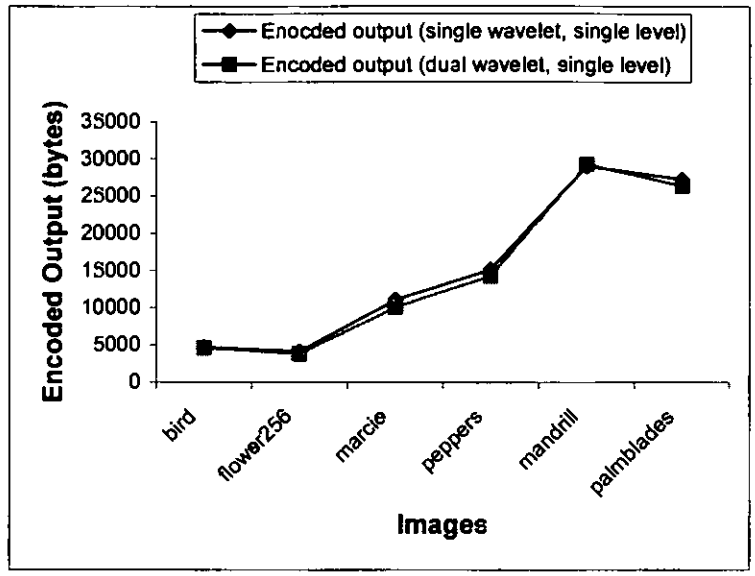


Figure 6.14 Comparison of conventional and segmented wavelet algorithms

The next enhancement was to apply the segmentation algorithm to the luminance image (Y) and the two chrominance images (C_bC_r). The smooth and non-smooth segments are compressed with the `bior4.4` and `haar` wavelet filters respectively. The results obtained are shown in Table 6.12.

Table 6.12 Coding performance of segment-based approach applied to Y and C_bC_r images

	bior4.4(4), haar(3), bior4.4(5), haar(3)	
	Encoded ouput	
Image	Total	PSNR
bird	4239	30.82
flower256	3146	25.75
marcie	9860	23.90
peppers	13977	17.20
mandrill	27533	19.29
palmblades	25622	13.16

Note: First two wavelets are used to decompose the segmented Y image at level 4 and 3.
The third and fourth wavelets are used to decompose the chrominance components at level 5 and 3.

The results in Table 6.12 show that there is a further reduction in the encoded output but there is also a significant loss in PSNR. In addition, there is also a significant colour distortion in the reconstructed image (See Figure 6.15). Hence, this particular approach will not be considered any further.

Original image *Image with colour distortion*

Figure 6.15 Effect of segmented method applied to all colour components

An alternative approach to further reduce the encoded output is to compress the segmented Y image with the `bior4.4` and `haar` wavelet filter and the non-segmented C_bC_r images using the `haar` wavelet filter. The results obtained are shown in Table 6.13.

The `bior4.4` and `haar` wavelet filters were chosen on their performance in relation to the colour planes they are to be used to compress. The `bior4.4` was shown to be effective for images with significant details present and the `haar` was shown to be best for images with distinctive sharp edges, example clipart (see chapter 5).

Table 6.13 Coding performance of segment-based approach dual level decomposition

	bior4.4(4), haar(3), haar(3)	
	Encoded ouput	
Image	**Total**	**PSNR**
bird	4096	36.23
flower256	3159	36.39
marcie	9088	30.44
peppers	13683	29.63
mandrill	28170	25.80
palmblades	26698	25.76

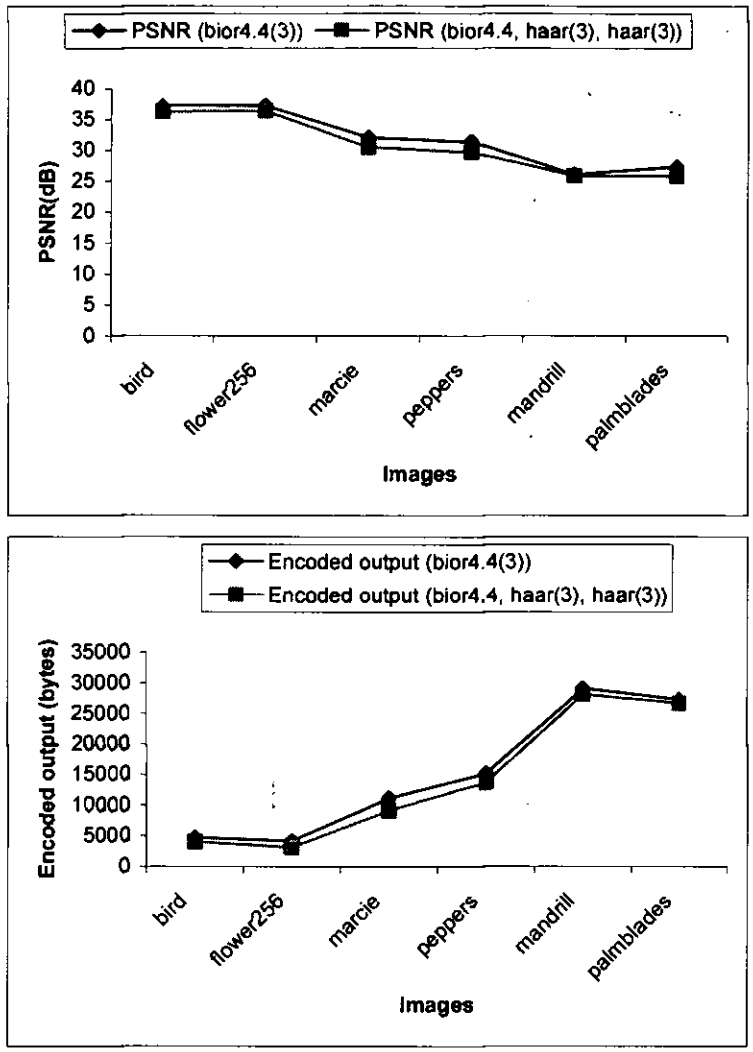


Figure 6.16 Comparison of two different approaches

Figure 6.16 gives a comparison of the results in Tables 6.10 and 6.13 and shows that the encoded output is further reduced using the dual wavelet approach and the PSNR loss is also reduced compared to the previous approach.

6.6. Summary

The results presented in section 6.2 represent the first stage in the development of an optimal algorithm for compressing still colour images for mobile telephones. The results in Table 6.3 show that different wavelets produce different levels of performance for the different images used as test cases. For example, considering the image of 'marcie', for best quality, the `bior6.8` wavelet should be chosen, offering a PSNR of 32.18. Equally, if the concern is for achieving the highest level of compression, the `db2` wavelet should be chosen, with a bpp of 0.40. However, the results of Figures 6.2, 6.3 and 6.4 show that the

same quality is not required across the whole image, giving rise to the possibility of selecting different wavelets for different parts of the image, based on local statistical properties.

The work in section 6.3 is concern with compressing colour images using different wavelet filters for the luminance component and chrominance components. The results, although confined to a small sample of images and range of wavelets, clearly demonstrate the potential gains that may be achieved by the application of either multiple wavelets or different levels of decomposition of the same wavelet. The image samples were chosen to give a good statistical spread of characteristics which gives confidence of the general applicability of this technique. The overall results for `bior4.4`,(3,5) are particularly impressive.

Section 6.4 presents a wavelet-based image compression scheme in which a grey-scale image is first partitioned into non-smooth and smooth segments and different wavelets are then applied using the 2-D DWT. The results reported in section 6.4 indicate that significant improvements in compression ratios can be achieved through the use of a combination of wavelets with little loss in the quality of an image. This could prove to be a very significant factor for mobile devices, where bandwidth costs can be high and small reductions in PSNR would have little, if any, impact. The method is extended to colour images where the luminance image is split into non-smooth and smooth segments and different wavelets are then applied to the segmented images. This method and the results are presented in section 6.5. The results in section 6.5.1 show that the coding performance is further improved by compressing the chrominance components with a different wavelet filter.

CHAPTER 7

A Hybrid Wavelet-based Compression System

7.1. Introduction

This chapter describes an image compression system called a hybrid wavelet-based compression system (HWCS), which is based on the assumption that viewers are more interested in the subject of interest (SOI) for a given image than the background. Indeed, the background could be a distraction from the subject of interest. In photography the background is sometime deliberately kept out of focus to reduce distraction from the subject of interest. For example, when photographing a child's birthday party, one of the subjects of interest is generally the child blowing out the candles on the cake and the background could be out of focus. Another example is nature photography where the subject of interest could be a lily or a deer. In these instances the subject of interest is sharp and the background is blurred.

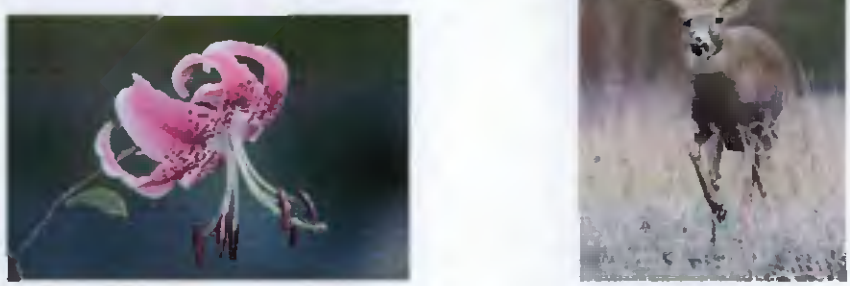


Figure 7.1 Examples of nature photography [source: Carnathan, 2004]

However, digital cameras and mobile telephone cameras tend to have small apertures, which mean the depth of field (DOF) is large. The result is that everything is in focus. When everything is in focus a lot of details are present and hence a lot of high frequencies, which in turn makes it more difficult to code the image. To make the image easier to code, it would be beneficial to have the background out of focus to reduce the amount of detail.

7.2. Hybrid Wavelet-based Compression System

Based on the earlier comparison made between a mobile telephone text message and a formal sentence construction, and also SOI, this research would like to present an alternative view to perceiving an image in the context of a mobile telephone. This thesis proposes to reduce the details in the background by averaging it and sending the background separately from the SOI. Figure 7.2 is a summary of the proposed HWCS:

```
1. Get image.
2. Crop image.
3. Create mask with cropped image.
4. Create background image with mask.
5. Derive the average background image.
6. Quantize and encode averaged image using Huffman entropy.
7. Compress cropped image.
8. Transmit the encoded compressed images and encoded averaged image.
9. Decode and dequantize averaged image.
10. Reconstruct decoded averaged image and compressed image.
11. Superimpose the compressed cropped image onto the averaged image
    to form the final image.
```

Figure 7.2 Overall algorithm

The SOI is a specific area within the image and this can be cropped. The cropping process is performed by selecting a specific area of the image, a sub-image, which is then cropped. The sub-image is compressed using a wavelet-based scheme as described in chapter six (section 6.3) and sent separately from the averaged background image. The final image is reconstructed by replacing the averaged background image pixels with the compressed cropped image.

It should be noted that the SOI used in this work is a very different approach to that used to define and process the region-of-interest (ROI) in the JPEG2000 standard. The ROI algorithm in JPEG2000 is designed to increase the quality of the selected area of the image rather than to reduce the resolution of the background.

7.2.1. The Background Image

A mask is created using the cropped sub-image. Initially the mask is black, which is of the same size as the copped image. The mask contains zeros for all pixels that are part of the cropped sub-image. This mask is used to replace the cropped portion from the original image to create the background image. Figure 7.3 illustrates this process.

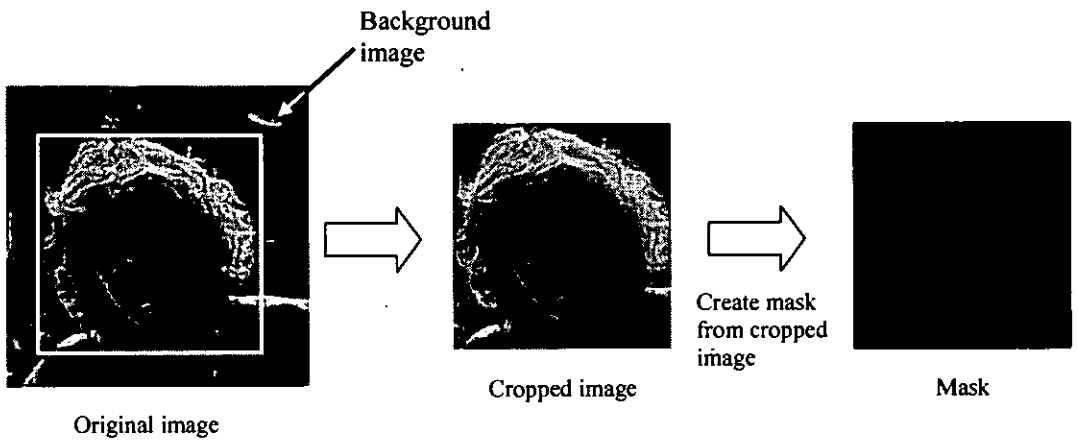


Figure 7.3Creating mask

However, the black mask produces a line artefact (Figure 7.4a) in the final reconstructed image. Figure 7.4(b) shows the same image with the line artefact removed. How the final reconstructed image is derived will be explained in a later section.

(a) *(b)*

Figure 7.4 Line artefact

To eliminate the artefact, the black mask is replaced with a different mask with an average value of the background image. The average value is computed from the background image using the following algorithm.

1. Create black mask from cropped image.
2. Replaced the cropped portion of the image with the black mask.
3. Convert the new image into a row vector.
4. Remove the zeros.
5. Compute the average value for generating the mask using the following

$$\tilde{x} = \frac{\sum I_{r,c}}{n}$$

where $\tilde{x}$ is the average value, n is the total no. of pixels and $\sum I_{r,c}$ is the sum of all pixels.

Figure 7.5 Algorithm for computing the average value of the mask

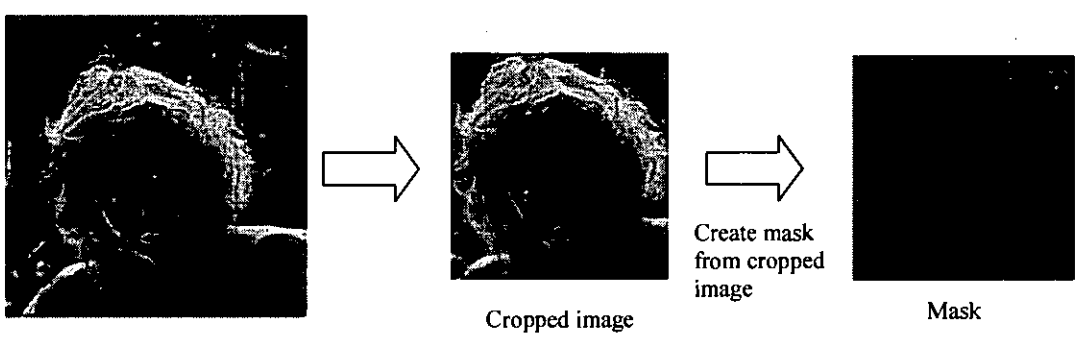
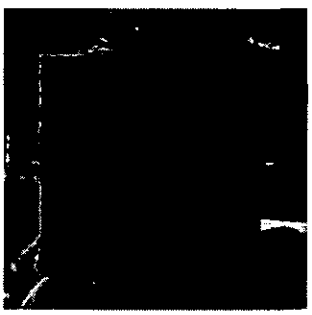


Figure 7.6 Result using the algorithm in Figure 7.5

The algorithm for creating the background image is shown in Figure 7.7(a) and the result is shown in Figure 7.7(b).

```
1. Define where to place the mask in the background image.
2. Perform the actual indexing to replace the background image pixels
   with the mask.
```

(a) Algorithm



(b) Background image with an averaged mask

Figure 7.7 Creating background image

7.2.2. Block-based Pixel Averaging

The section describes the details of the process used to derive the average background image utilizing a block-based operation. Block-based operations involve processing an image in sections called blocks, rather than processing the entire image hence the name block-based operations. This is illustrated by the following figure:

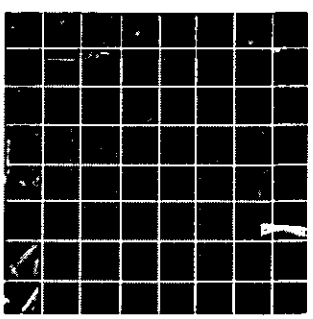


Figure 7.8 Block-based operations

The following figure (Figure 7.9) is used to illustrate how the averaged background image is derived. Assuming the background image is defined by a set of pixel values $I_{r,c}$, where *r,c* is the pixel coordinate.

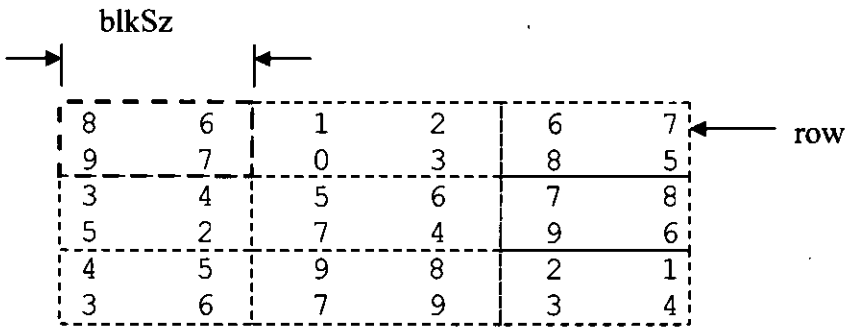


Figure 7.9 Background image matrix

The background image is divided into $m-$by$-n$ blocks, which is manually determined by the user. The block sizes are 4, 8, 16, and 32 were used in this research. The block size can however also be determined automatically by another Matlab function and this function is described later in this chapter.

The average of each block is computed using:

$$\tilde{a} = \frac{\sum_{0}^{m}\sum_{0}^{n} I_{r,c}}{m * n}$$

where $\tilde{a}$ is averaged pixel value, and $m * n$ is the dimension of the image. The result is an averaged image, which is smaller than the input matrix.

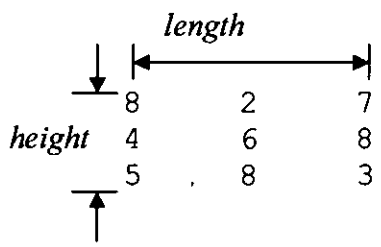


Figure 7.10 Averaged matrix

The output matrix is then quantized by a quantizer described in section 6.3. The output of the quantizer is fed into a Huffman entropy encoder. The resultant output from the encoder

is transmitted with the compressed part of the image. Both the quantizing and entropy encoding processes modify the pixel values in order to save memory and shorten transmission time.

7.2.3. Reconstruction of Background Image

At the receiver end the encoded output is then decoded. The decoded output is used to reconstruct the background image. The background image is reconstructed using the averaged value in each block. The resultant background image is an averaged value image and it is referred to as the averaged background image. The reconstruction algorithm is as follows:

```
1. Initialization
    Set the element and its position to 1.
    Initialize start of block and end of block value.
    Set eTrack to 1.

2. Reconstruction:
    while Height <= height of averaged image

      while row <= height of block

            while eTrack <= length of averaged image

                for m = start of block : end of block
                    read element of block and create a new list;
                endfor

                update start of block marker;
                update end of block marker;
                point to the next element in block;
                store element in buffer;
                update bTrack;

             endwhile
             update row by 1;
             restore initial element;
             restore initial row;

      endwhile
       recall buffer;
       Refresh eFirst with buffer;
       restore initial row;
       update Height by 1;

   endwhile
```

Figure 7.11 Reconstruction Algorithm

7.3. Reconstruction of the Final Image

The final image is reconstructed using the following algorithm:

```
1. Define where to place the compressed cropped image in the averaged
   background image.
2. Perform the actual indexing to replace the averaged background
   image pixels with the compressed cropped image.
```

Figure 7.12 Final image reconstruction algorithm

7.4. Results for Grey-scale Images

The following set of results (Table 7.1) gives a comparison of the output from the encoder for a single wavelet (`bior4.4`) and the HWCS. The number of bytes coming from the encoder has been used to provide an absolute measure of performance.

Table 7.1 Encoder output without quantizer

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
gbird	2022	2811	1778	1573	1396
gmarcie	6849	6543	4916	4516	4345
gchiliq01	9238	8721	7224	6843	6680
gmandrill	19125	13603	12187	11872	11685
gchiliq02	7685	7571	6048	5648	5471
gflower256	2731	2731	1467	1112	923

The results from Table 7.1 indicate that significant reduction in the encoder output can be achieved through the HWCS. This could prove to be an important factor for mobile telephone service providers, where transmission time may be significantly reduced. In addition the amount of storage required for each image is also significantly reduced, which means that more images can be stored on the mobile telephones.

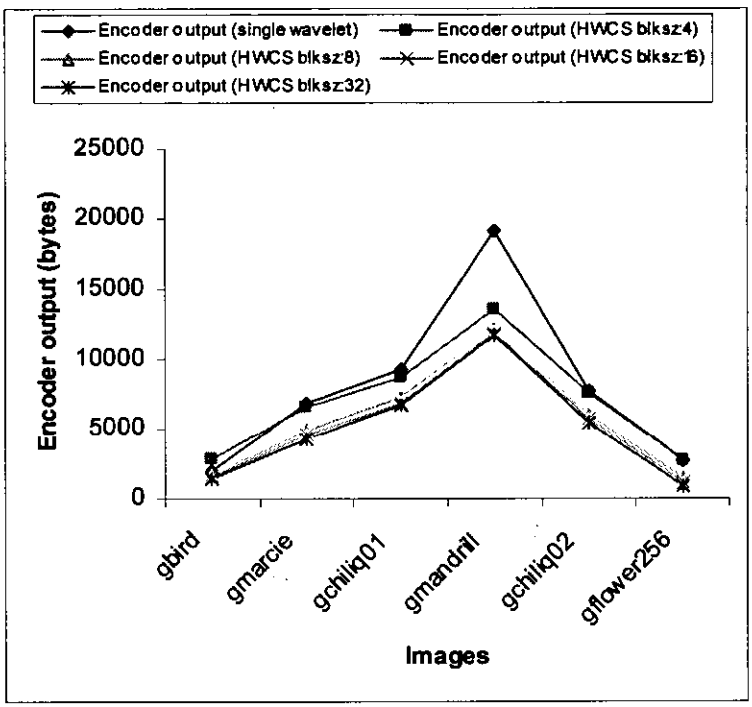
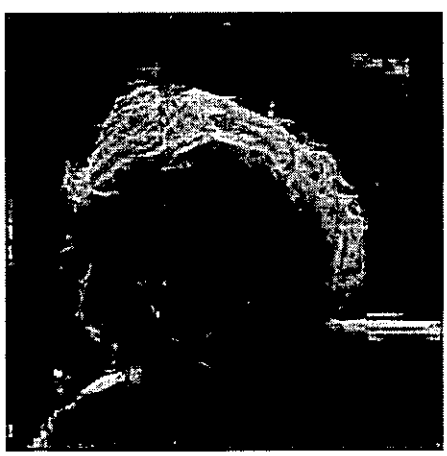
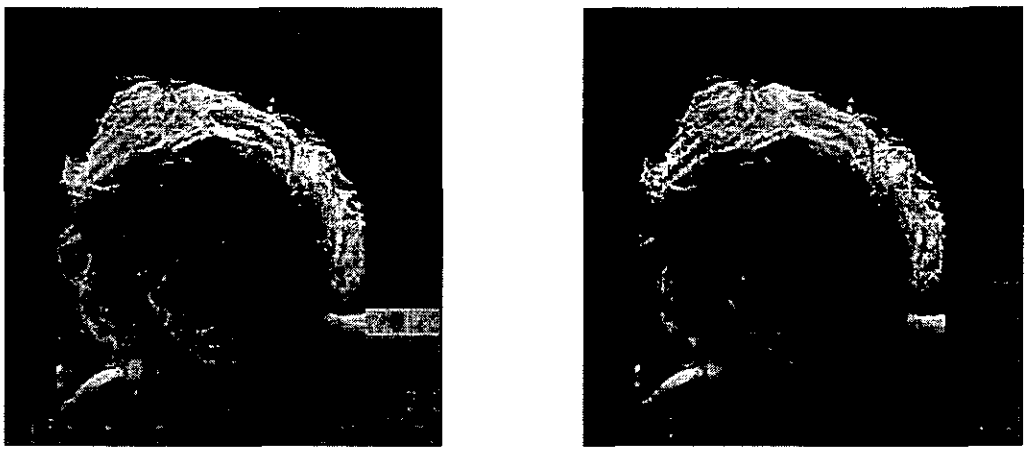


Figure 7.13 Graphical compression of encoder output



Block size: 4

Block size: 8



Block size: 16 Block size: 32

Figure 7.14 Reconstructed grey-scale images at different block sizes

To achieve a smaller encoder output, the average image is quantized prior to encoding. Table 7.2 presents the results.

Table 7.2 Encoder output with quantizer

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
gbird	2022	2218	1585	1424	1380
gmarcie	6849	5586	4662	4400	4327
gchiliq01	9238	7921	7017	6742	6661
gmandrill	19125	12896	11953	11715	11654
gchiliq02	7685	6776	5801	5530	5452
gflower256	2731	1710	1158	969	891

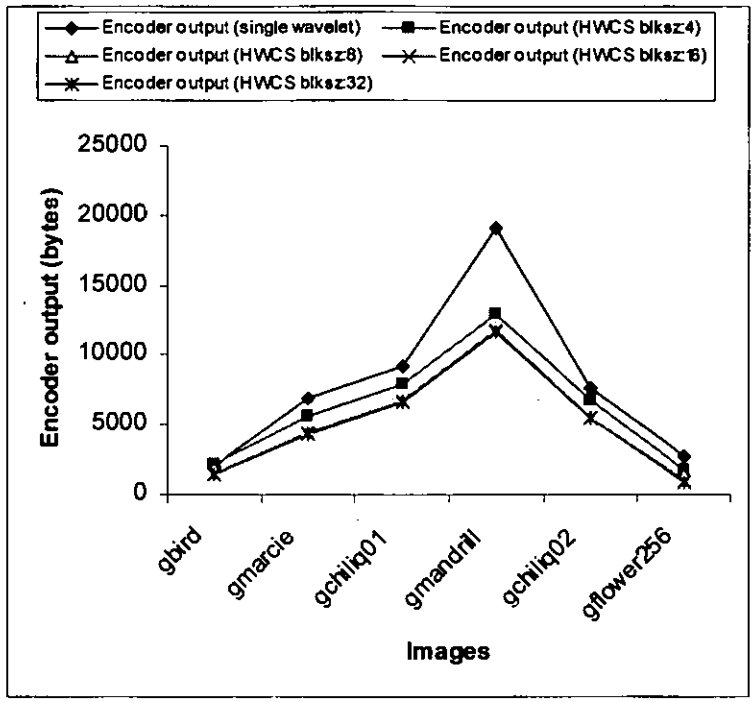


Figure 7.15 Graphical comparison of encoder output with quantizer

7.5. Block Size Selection

Up until now the block size is selected manually. This section will describe an automated block selection process. The block size is determined by determining the level of activity of the background image with the cropped portion replaced by a black mask. This background image is created using the process described in section 7.3.

Figure 7.16 Background image with a black mask

A block-based Fast-Fourier transform (FFT) method described in section 6.2 is used to determine the level of activity of the background image. The level of activity is classified as either low, moderate, or high.

Having determined the level of activity, the following algorithm is used to select the block size.

```
if Low > Moderate
   if (Moderate + High) >= 60
      Select Moderate;
      Set Flag to 0;
   else
      Select Low;
      Set Flag to 0;
   endifelse
endif

if Flag not 0
   if Moderate > High
      if H  >=`35
         Select High;
         Set Flag to 0;
      endif
      if Flag not 0
         Select Moderate
      endif
   else
      Select High
   endif
endif

if Select = Low
   Set blksize = 16
endif
if Select = Moderate
   Set blksize = 8
endif
if Select = High
   Set blksize = 4
endif
```

Figure 7.17 Block size selection algorithm

7.6. Compressing Colour Images Using HWCS

The HWCS described in earlier sections is used to compress grey-scale images. In this section the HWCS is extended to compress colour images. There are two parts to the proposed scheme. Part one consists of applying the HWCS, described in the above sections, to the luminance (Y) image. The second part consists of applying the 2D DWT to compress the C_bC_r components as described in section 6.3.

7.6.1. HWCS Results

The following set of results in Table 7.3 gives a comparison of the output from the encoder for a single wavelet (`bior4.4`) and the HWCS. The total number of bytes coming from the encoder has been used to provide an absolute measure of performance.

Table 7.3 Colour encoder output

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
bird	4730	5487	4478	4242	4098
marcie	11066	10755	9135	8731	8560
chiliq01	14182	13693	12195	11810	11648
mandrill	29063	23510	22087	21775	21588
chiliq02	10934	10832	9301	8904	8727
flower256	4161	4668	3668	3432	3231

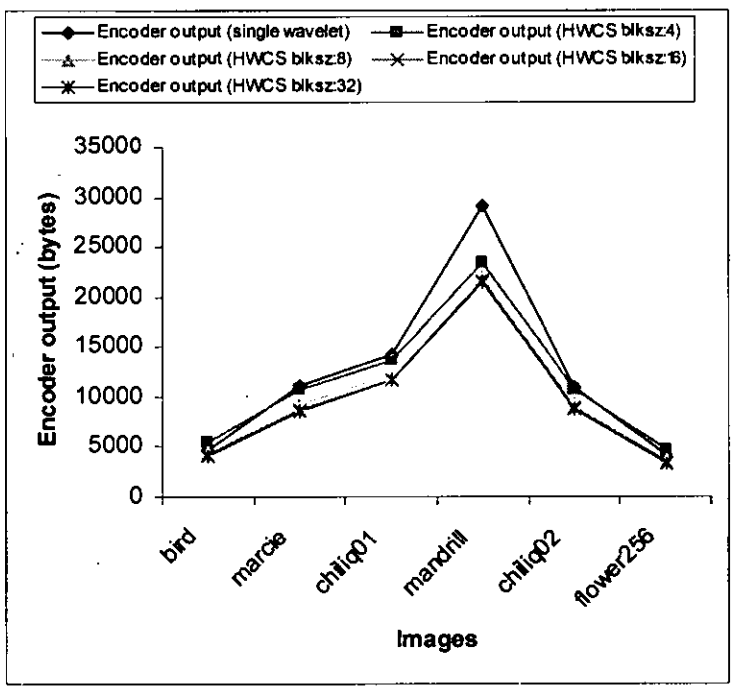


Figure 7.18 Graphical comparison of colour encoder output

As with the grey-scale images, the results from Table 7.3 indicate that significant reduction in encoder output for colour images can be achieved through the HWCS. Again this could prove to be an important factor for mobile telephone service providers, where transmission time may be significantly reduced. The following are samples of images compressed by the HWCS.

Block size: 4

Block size: 8

Block size: 16

Block size: 32

Figure 7.19 Reconstructed colour images with different block sizes

7.7. Compressing All Colour Components Using HWCS

In the next stage the HWCS was applied to all three colour components to achieve a smaller encoder output. Table 7.4 presents the set of results.

Table 7.4 Encoder output with HWCS applied to all three colour components

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)
bird	4730	5121	3420
marcie	11066	10529	7836
chiliq01	14182	13913	11234
mandrill	29063	21762	19364
chiliq02	10934	11411	8848
flower256	4161	4073	2504

The following are samples of images compressed by this approach:

Figure 7.20 Reconstructed images with HWCS applied to all three components

The results in Table 7.4 show that the applying the HWCS to all three colour components can effectively further reduce the encoder output. However, as can be observed from Figure 7.20, the images have been discoloured and this is not acceptable. Therefore this approach will not be considered any further.

7.8. Using a Quantizer

This section explores an alternative approach to further reduce the encoder output. To achieve smaller encoder output the average image can be quantized prior to encoding. Table 7.5 presents the set of results.

Table 7.5 Colour encoder output (using quantizer)

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
bird	4730	4884	4294	4126	4082
marcie	11066	9815	8877	8615	8542
chiliq01	14182	12891	11981	11710	11629
mandrill	29063	22798	21856	21618	21557
chiliq02	10934	10022	9051	8785	8708
flower256	4161	4158	3447	3257	3207

Figure 7.21(a) is the output image produced by the HWCS without the average image quantized prior to encoding and Figure 7.21(b) is the output image produced by the HWCS with the average image quantized prior to encoding. Figures 7.21(a) and 7.21(b) show that this approach produces a further reduction in the encoder output while maintaining the colour of image.

(a) Without quantizer *(b) With quantizer*

Figure 7.21 Comparing images

7.9. Using Dual Wavelet and Higher Level of Decomposition

This section explores another approach to reduce the encoder output by using dual wavelets to compress the cropped image. The proposed approach applies the HWCS to the luminance image. The `bior4.4` wavelet filter is used to decompose the cropped luminance image and a `haar` wavelet filter is used to decompose the chrominance part (C_bC_r) of the image. In addition, the chrominance part is also decomposed at one level higher than the luminance part. The results are tabulated in Table 7.6.

Table 7.6 Results for dual wavelet and level approach

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
bird	4730	4320	3311	4126	4082
marcie	11066	9040	7420	8615	8542
chiliq01	14182	12829	11331	11710	11629
mandrill	29063	22077	20654	21618	21557
chiliq02	10425	10425	8894	8785	8708
flower256	4161	3723	2720	3257	3207

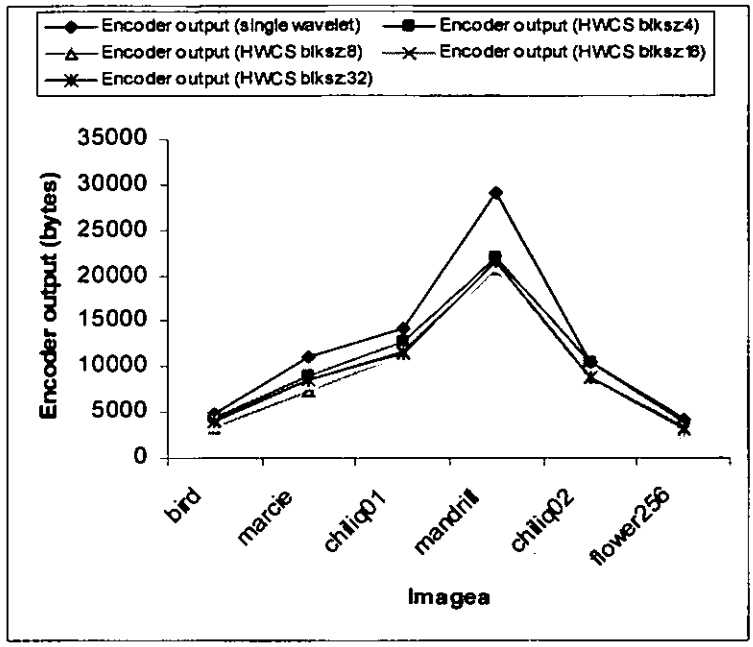


Figure 7.22 Graphical comparison of encoder output using dual wavelet and level approach

The results in Table 7.6 and Figure 7.22 clearly show that the HWCS dual wavelet and level approach can reduced the encoder output significantly compared to HWCS quantizer approach.

Further experimental work shows that different orthogonal wavelets can be used to decompose the chrominance components that can further reduce the encoder output. The following table shows the results:

Table 7.7 Effect of using different orthogonal wavelet on C_bC_r components

	bior4.4(3)		haar(3)		db3(3)		sym2(3)	
Imaga	CbEncoded	CrEncoded	CbEncoded	CrEncoded	CbEncoded	CrEncoded	CbEncoded	CrEncoded
8mile01_256	1844	1699	**1861**	**1418**	1771	1518	1668	1463
aerial01_256	1785	1971	1337	**1633**	1646	1793	1566	1675
peppers	3331	3844	2716	3930	2931	3556	**2878**	**3424**
lenacolor	2390	2202	2312	2082	2230	2131	**2208**	**2053**
palmblades	9230	2731	*10103*	2522	9329	2642	**9178**	2565
boy	1884	2018	1626	1701	1645	1877	**1582**	1810
bird	1521	1179	**1208**	**1003**	1331	1070	1268	1132
marcie	2156	2063	**1719**	**1602**	1967	1837	1766	1695
chiliq01	2245	2715	2085	2633	1996	2504	**1927**	**2470**
mandrill	5777	4151	5308	4151	5406	**3976**	**5281**	3998
chiliq02	1705	1555	1626	1671	1648	**1536**	**1548**	1550
flower256	1082	1337	**927**	**926**	936	1044	937	979

7.10. Applying HWCS to Mobile Telephones Images

As indicated in the earlier part of this thesis, the main concern of this research is picture messaging using mobile telephones. This section will discuss the work of applying the HWCS to images used on mobile telephones. This is done by resizing the image to a fit the screen of a commercially available mobile telephone. The screen size in this instance is 128 x 128 pixels. It should be noted up to this stage of the research, the images used are 256 x 256 pixels.

Having resized the image at source the HWCS with dual and level approach is applied to the resized image. The following is a set of results (Table 7.7) using the mentioned approach.

Table 7.8 Compressing 128 x 128 images using HWCS

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:4)	Encoder output (HWCS blksz:8)	Encoder output (HWCS blksz:16)	Encoder output (HWCS blksz:32)
bird	2168	1870	1597	1453	1417
marcie	5090	3695	3272	3100	3054
chiliq01	5735	5055	4659	4497	4454
mandrill	10241	7794	7461	7275	7228
chiliq02	5349	4670	4249	4071	4027
flower256	1763	1442	1183	982	940

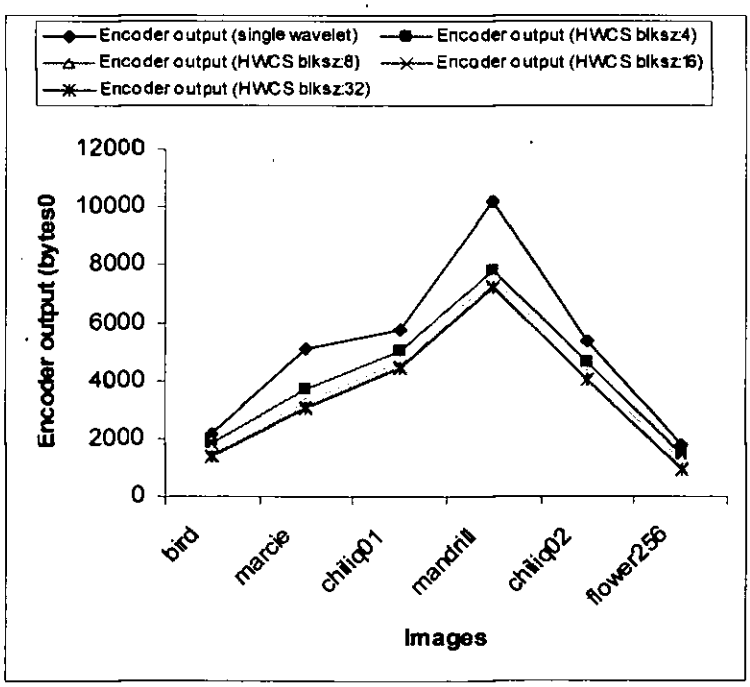


Figure 7.23 Graphical comparison of encoder output for 128 x 128 images

Figure 7.24 show four colour images resized and compressed using HWCS. They are displayed in context of the mobile telephone environment.

Block size: 4

Block size: 8

Block size: 16 Block size: 32

Figure 7.24 Sample images

In order to provide a direct comparison, the PSNR for the single wavelet method was set to the same level as that achieved by the HWCS method. The number of bytes produeed by the eneoder was recomputed. The results are shown in Figure 7.25.

Image	Encoder output (single wavelet)	Encoder output (HWCS blksz:8)
bird	1083	1597
marcie	2492	3272
chiliq01	2740	4659
mandrill	6479	7461
chiliq02	2695	4249
flower256	1028	1183

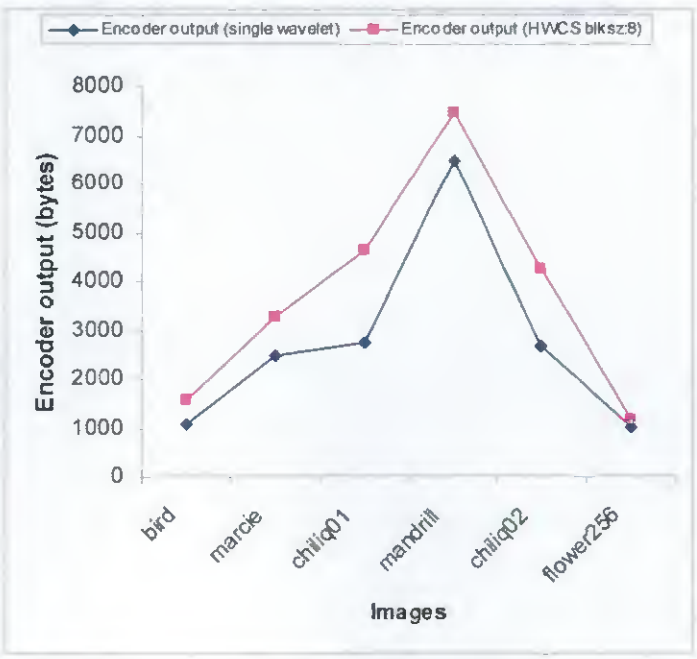


Figure 7.25 Comparison of the single wavelet and the HWCS method for fixed PSNR

The results indicate that the encoder output produced by the single wavelet method is lower than the HWCS method. However, the visual quality of the image produced by the single wavelet method is very poor. The encoder output for the 'bird' image is presented in Figure 7.26 for *(a)* the HWCS method, and *(b)* the single wavelet method both set at 26.06dB.

(a) HWCS *(b) Single wavelet*

Figure 7.26 Comparison of encoder output for fixed PSNR value

7.11. Fidelity Criteria

Fidelity criteria measures the difference between the original image and the reconstructed (decompressed) image. It is divided into objective fidelity criteria and subjective fidelity criteria.

The objective fidelity criteria originates from signal processing. An example of objective fidelity criteria is the root-mean-square (RMS) error, which is defined as:

$$e_{rms} = \sqrt{\left[\frac{1}{m*n}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}\left[\hat{I}(x,y) - I(x,y)\right]\right]} \tag{7.1}$$

where e_{rms} is the RMS error (RMSE), $m*n$ is the size of reconstructed image, $\hat{I}(x,y)$ is the reconstructed image and $I(x,y)$ is the original image [Efford, 2000, pp.301]. A closely related objective fidelity criterion is the PSNR.

The PSNR measure is used as an estimate of the quality of a reconstructed image compared with an original image. It is computed using the MSE defined in equation 5.2 in section 5.4.3. Equation 5.2 is only an approximation of $I(x,y)$ and therefore it can be argued that PSNR is only an approximation. PSNR is a good measure for comparing a reconstructed image with the original image, but comparing visual quality of images using PSNR is meaningless. For example an image with a 15 dB PSNR may look better than an image with a 25 dB PSNR.

MSE and PSNR are convenient mechanisms for evaluating information loss and they are widely used in many applications, in particular in image coding. One reason why they are so widely used is because they are easy to calculate and usually have low computational complexity. However, they do not correlate well with the human perception of image quality [Efford, 2000, pp.301; Watson, 1993]. Measuring image quality by subjective evaluations of human observers often is more appropriate. Furthermore, most reconstructed images are ultimately viewed by humans. For this reason, this thesis argues that subjective fidelity criteria based on human perception is more appropriate for evaluating image quality.

7.12. Subjective Fidelity Criteria

Subjective fidelity criteria can be divided into three categories:

1. Impairment
2. Quality
3. Comparison

Impairment tests score the test images in terms of how bad they are. Quality tests rate the test images in terms of how good they are and comparison tests evaluate test images on a side-by-side basis. Table 7.9 provides an example of scoring scales for the three types of subjective fidelity measures.

Table 7.9 Subjective fidelity scoring scales [Umbaugh, 1998, pp.242]

Impairment	Quality	Comparison
5 – Imperceptible	A – Excellent	+2 much better
4 – Perceptible, not annoying	B – Good	+1 better
3 – Somewhat annoying	C – Fair	0 the same
2 – Severely annoying	D – Poor	-1 worse
1 – Unusable	E – Bad	-2 much worse

The definitions in Table 7.9 are vague and therefore another scoring scale is needed. Table 7.10 provides another example of one possible absolute rating scale for evaluating television images.

Table 7.10 Television allocation study organization rating scale [Frendendall and Behrend, 1960]

Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality.
2	Fine	An image of high quality. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very poor image, but still acceptable. Objectionable artefacts are highly visible.
6	Unusable	An image so bad that it is unacceptable.

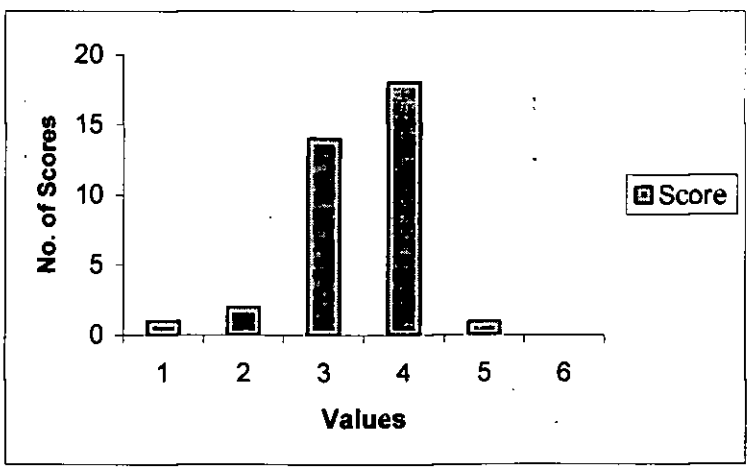
The Table 7.10 scoring scale is more suitable for evaluating still images but it needs to be modified for the purpose of this research. A modified version of Frendendall and Behrend [1960] is presented below:

Table 7.11 Alternate subjective fidelity scoring scales

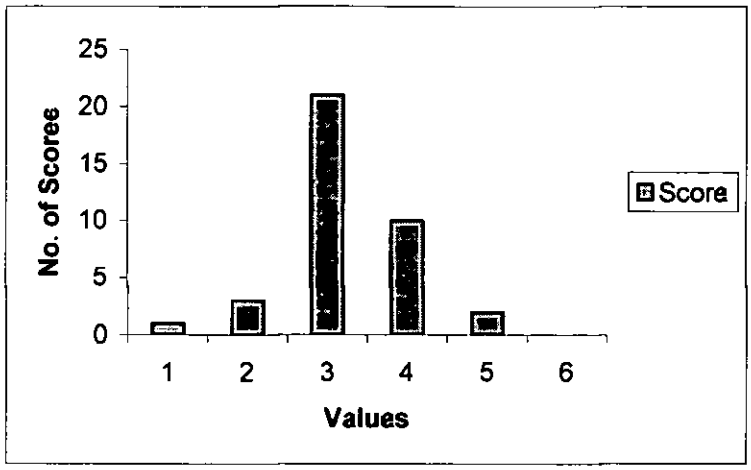
Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very of poor image, but still acceptable. Objectionable artefacts are definitely present.
6	Unusable	An image so bad that you do not want it.

7.13. Image Evaluation

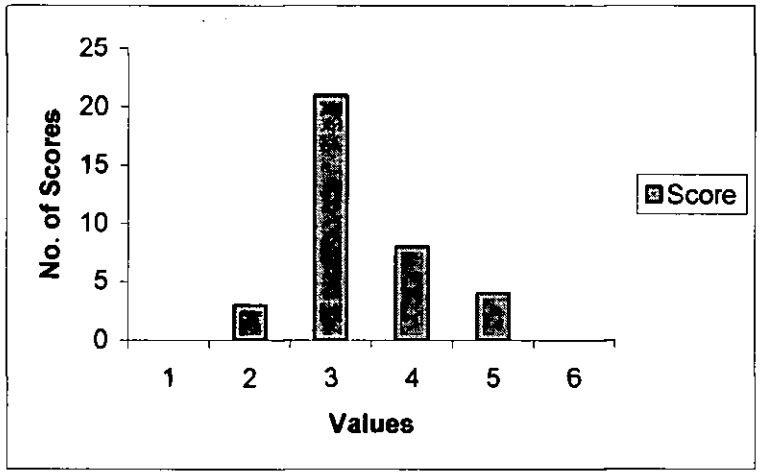
A subjective evaluation of the visual quality of the four images in Figure 7.24 was carried out with thirty-six postgraduate students using the questionnaire in Appendix 4. The postgraduate students are enrolled on a Masters of Science in Computer Networks. This group of students were chosen because they are mobile telephone users or potential mobile telephone users who are most likely to use picture messaging. Each student was asked to rate each image using the subjective fidelity scoring scales displayed in Table 7.10. The results are shown in Figure 7.27.



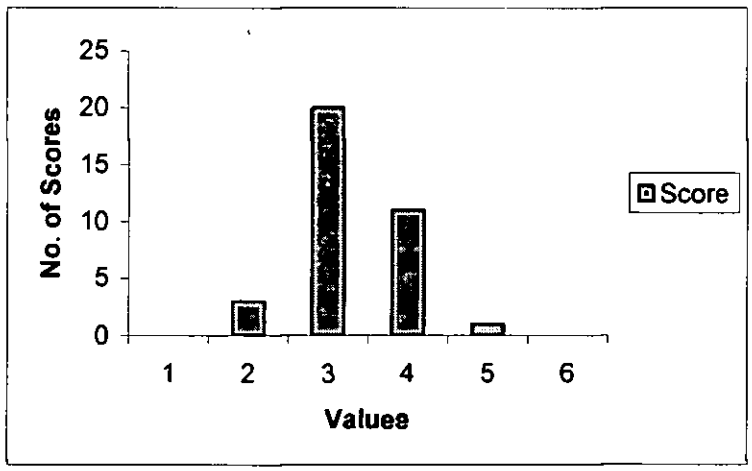
(a) Block size 4



(b) Block size 8



(c) Block size 16



(d) Block size 32

Figure 7.27 Subjective fidelity scoring results

The results show that the majority of the respondents have rated the four images as either *passable* (3) or *marginal* (4). For images compressed using 8x8, 16x16 and 32x32 block sizes the swing is towards passable. This is an unexpected and interesting but potentially very important result. Although the sample size was small, the larger block sizes, that is, 8x8, 16x16, and 32x32 scored better than the smaller one (4x4). Figure 7.27(a) shows that the majority of the respondents score a rating of 4 (marginal) for the 4x4 block size whereas the larger block sizes produced a majority score of 3 (passable), which is a better score. The most significant result is that the largest block size of 32x32 produced scores that are broadly comparable to the finer block sizes. This means that a larger block size may be used producing better compression, while providing image of acceptable quality.

By inspecting and comparing the 4x4 and 32x32 images in Figure 7.28 it is possible to offer an explanation why the respondents of the subjective evaluation favour the bigger block size images. The two images are repeated here for convenience.

(a) 4x4 block size *(b) 32x32 block size*

Figure 7.28 Comparing the 4x4 and the 32x32 block size image

It can be observed that the 4x4 block size image has more jagged edges than the 32x32 block size image. A close up view of part of the 32x32 block image reveals that some details have emerged despite the fact that the image has been subjected to a coarse averaging algorithm. This thesis argues that this is possible because the averaging algorithm was only applied to the Y component and not to the chrominance components. Hence, during the colour inverse transformation process, the details from the chrominance components have emerged.

Figure 7.29 Effect of chrominance components

Figure 7.30 is a grey-scale version of Figure 7.29 and it shows that the details visible in the colour images are not visible in the grey-scale image.

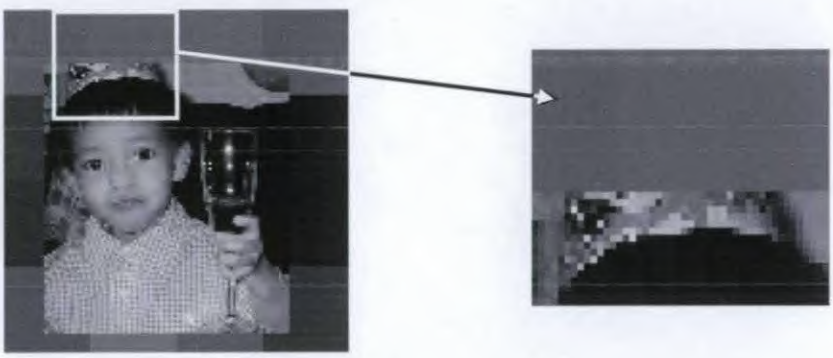


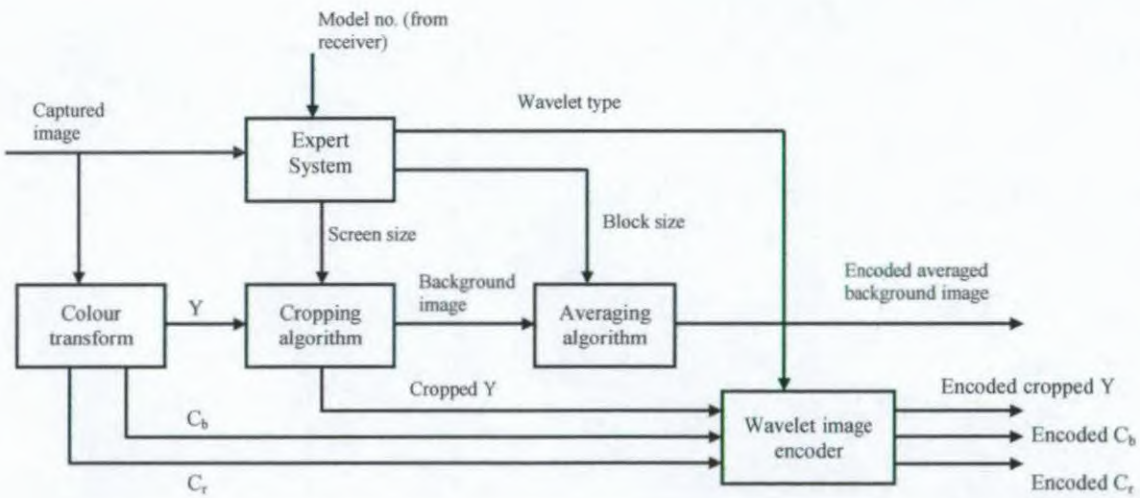
Figure 7.30 Grey-scale version of the image in Figure 7.29

The next stage in the investigation is to apply the averaging algorithm to the Y, C_b, and C_r components. The result is shown in Figure 7.31 and it shows that the details that are visible in Figure 7.29 colour are not visible here. Hence, this thesis concludes that this supports the postulate that the chrominance components contributed to the details that are visible in the colour image produced by the HWCS.

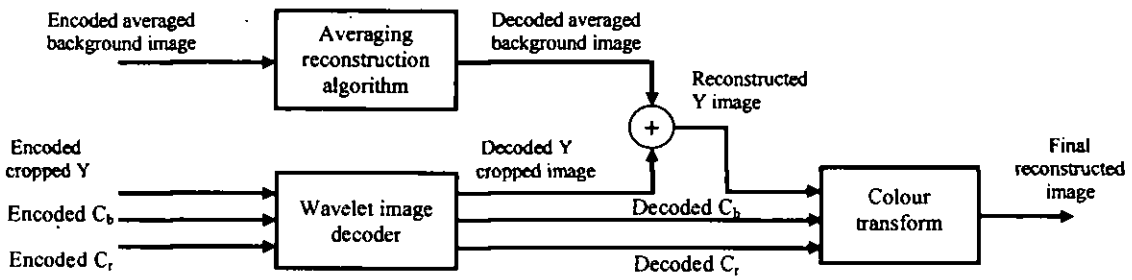
Figure 7.31 Averaging applied to the YC_bC_r components

7.14. A Proposed Wavelet-based Compression System for Mobile Telephones

Using the results from chapters five, six, and seven a wavelet-based compression system for mobile telephones is proposed. The block diagram of the system is shown in Figure 7.32(a) and Figure 7.32(b).



(a) Encoder



(b) Decoder

Figure 7.32 Proposed Wavelet-based Compression System for Mobile Telephones

The expert system in Figure 7.32(a) is a rule-based system. Its function is to select a screen size suitable for the receiving device. This is accomplished by the transmitting device interrogating the receiving device for the model number. When the model number is received, the transmitting device will search its database for the appropriate screen size. The second function of the expert system is to select the best wavelet by analysing the captured image using a block-based FFT analyser described in section 6.2. The result from the block-based FFT analyser is then used select the appropriate wavelet. The third function of the expert system is to select the appropriate block size for the averaging algorithm.

The cropping algorithm cropped the Y component from the YCbCr image which had been transformed from the captured RGB input image. The cropping algorithm crops a specific area of interest of the image, which is selected by the user. The remainder of the image, that is, the background image is then averaged. The averaging algorithm averages the background image utilizing a block-based operation, which is described in section 7.2.2. The wavelet image coder is a two-dimensional wavelet transform discussed in section 3.3.5.

On the receiving device, the encoded averaged background image is reconstructed by the averaging reconstructing algorithm. The resultant decoded averaged background is then combined with the decoded Y image to produce the final Y image. Finally, an inverse

mapping is carried out to convert the Y, C_b, and C_r colour components to R, G, and B components.

The proposed system was simulated via Matlab using the algorithms developed in this research. For simulation purposes, a set of natural, synthetic and clipart images were fed into the system separately. The system was able to identify the images and choose the best wavelet to compress the image concerned. The simulation shows that the proposed system was able to perform the cropping and averaging algorithm effectively as predicted. Further simulation also showed that given the model number of the receiving device, the system was able to select the appropriate screen size.

7.15. Summary

This chapter presented a hybrid wavelet-based compression system (HWCS), which is based on the assumption that viewers are more interested in the subject of interest (SOI) for a given image than the background. The HWCS reduces the details of the background by averaging it and sending the background separately from the SOI. As the main concern of this research is colour picture messaging using mobile telephones, the HWCS was extended to colour images suitable for use on mobile telephones. The HWCS is discussed in detail in this chapter. The results of the image evaluation reported in section 7.13. Although the sample is small, the results demonstrate that the algorithm has produced images that are visually acceptable for picture messaging, while improving overall compression over existing methods.

CHAPTER 8

Discussion and Conclusions

8.1. Introduction

In the introduction to this thesis the problems associated with transmitting colour images were discussed in detail. The discussions pointed out that although significant improvements have been made in media storage technology and transmission media, the problem of low air bandwidth that can affect Internet access via mobile telephones, remains a concern. Therefore, this thesis argues that this justifies putting efforts into research in image compression for mobile telephones.

8.2. Small Screen Factor

The discussions in section I.3 highlighted the problem of the disadvantage of the small screen of mobile telephone. However, the discussion also pointed out that the small screen factor offers an opportunity for a trade-off between image quality (PSNR) and image size (bytes transmitted). Therefore, different wavelet-based compression schemes that offer a trade off between the image quality (PSNR) and image size (bytes transmitted) were investigated. What was envisaged was a compression system that would be able to select an optimal compression method to meet both the demands specified by the user that were matched to their mobile telephone screen. The main purpose of this thesis was to analyse the use of wavelet-based image compression methods and then developed new techniques suitable for application to mobile devices, with particular reference to picture messaging on mobile telephones.

8.3. Bandwidth Problem

Another problem that is highlighted in the section I.3 is the air bandwidth bottleneck. This problem is expected to remain as the growth of Internet traffic using the air bandwidth is likely to continue, as the number of mobile devices used to access the Internet is likely to increase. In addition, the Internet protocols like TCP/IP and HTTP, which come with a lot of overheads, are not suitable for wireless telecommunication. Therefore, effective data compression techniques for mobile telephones are essential for transmitting and storing digital images.

8.4. Choosing the Best Wavelet

Chapter five considers the characteristics of colour images and how they influence coding performance. The results from the statistical study showed that it is image gradient and spatial frequency that has a strong correlation with PSNR, CR and bpp. The magnitude of the relation between the variables, that is, image gradient and the coding performance and spatial frequency and coding performance is strong. In other words, one variable could be predicted based on the other. However, this result is only a statistical association and cannot conclusively prove causality. A more detail study is needed to establish what is the cause of this association.

The statistical results from chapter five showed that it is difficult to distinguish between a natural image and synthetic image. However, it was observed that first-order entropy, *H(1)*, of the clipart images tended to be less than 5. This could be used in an expert system to identify a clipart image. The simulation results reported in section 7.14, indicate that it is possible to identify clipart images using *H(1)* as a identifying criteria.

Another concern of chapter five was to establish, which wavelet filter was best suited to compress the different types of image used in this research. Hence, a study was conducted and to address this concern. The results of the study show that that the best combination of PSNR and CR was produced by the '`bior4.4`' (9/7 tap) wavelet filter and the `haar` wavelet filter outperforms the other wavelets for clipart images.

8.5. Wavelet-based Image Compression Schemes

Motivated by the success and advantages of the DWT as a compression technique, this thesis has examined and implemented three main wavelet-based compression schemes with a focus on a suitable wavelet-based compression method for mobile applications. The details of these three algorithms are reported in chapters 6 and 7.

The first algorithm is a dual wavelet compression algorithm. This algorithm is a modified conventional wavelet compression method, which uses different wavelet filters to decompose the luminance and chrominance components separately. In addition, different levels of decomposition can also be applied to each component separately. It also avoids the use of advanced quantization schemes like EZW to achieve reduced encoder output. The need to keep complexity to a minimum is an important factor for mobile applications.

The results of the evaluation of this algorithm indicate that there is a possibility of a triple wavelet compression system, where a different wavelet is used for each of the colour components, that is, Y, C_b, and C_r separately. The evaluation results also clearly show that potential gains may be achieved by the application of either multiple wavelets or different levels of decomposition of the same wavelet.

The second method is a segmented wavelet-based algorithm, which segments an image into its smooth and non-smooth parts. A different wavelet filter is then applied to the segmented parts of the image. The results show that significant improvements in compression ratio can be achieved through the use of a combination of wavelets with little loss in the quality of a grey image. When the method was extended to compress colour images the results showed that the coding performance could be further improved by compressing the segmented chrominance components with a different wavelet filter and different level of decomposition.

Finally, the third algorithm was the HWCS, where the subject of interest was cropped and then compressed using a wavelet-based method. The details of the background are reduced by averaging and sending the background separately from the compressed subject of interest. The final image is reconstructed by replacing the averaged background image pixels with the compressed cropped image. The HWCS was extended to compress colour images, where the HWS is applied to the luminance (Y) image. The C_bC_r components are compressed using the 2D DWT.

For each algorithm described the results show that encoder output can be effectively reduced when compared to conventional wavelet-based compression schemes. The subjective evaluation of the images produced by the HWCS show that the images are visually acceptable for picture messaging, while improving overall compression.

8.6. Contributions

This section describes the contributions made by this thesis. The contributions are classified into main contributions based on the work carried out in this thesis and contributions based on previous work. This classification is, of course, subjective and represents the author's opinion. The references in parenthesis refer to the corresponding sections in the thesis.

8.6.1. Main Contributions from this Thesis

i. A hybrid wavelet-based compression scheme for mobile devices was designed, developed and proven experimentally (chapter 7).

ii. A segmentation wavelet-based compression scheme for grey-scale and colour images was designed and tested (section 6.4 and 6.5).

iii. A fully automated wavelet-based compression system for mobile applications has been proposed and simulated (section 7.24).

8.6.2. Other Contributions

i. Created a block-based FFT analyser for analysing frequency content of an image (section 6.2).

ii. Developed a dual wavelet compression scheme for compressing colour images (section 6.3).

iii. Established that the `haar` wavelet filter outperforms the other wavelets for clipart images (section 5.4.3).

iv. Shown that the first-order entropy, *H(1)*, can be used distinguish clipart from natural and synthetic image (section 5.5).

v. Shown from a statistical study that the image gradient and spatial frequency of colour images has a strong linear relationship with PSNR, CR and bpp (section 5.6 and 5.7).

vi. Shown, with the aid of experimental work, that different orthogonal wavelets can be used to decompose the chrominance components that can further reduce the encoder output (section 7.9).

8.7. Future Work

Based on the work reported in this thesis, this section puts forward some ideas for future work.

8.7.1. Cause-Effect Study

As discussed in section 8.4, the statistical study in chapter 5 shows that there is strong statistical association between image gradient and the coding performance and spatial frequency and coding performance. However, the cause-effect relationship was not established. Therefore, this thesis would suggest a detailed investigation into the cause-

effect relationship would be the next logical direction. The cause-effect study could help better understand which image characteristics influence the coding performance of a wavelet-based image compression system. Hence, an adaptive wavelet-based compression system would be feasible based on the image characteristics. The adaptive wavelet-based compression system could be a neural-network, which would select the best wavelet based on image characteristics.

The study could also include a more detailed investigation as to why the zero-order entropy measure in the RGB colour space has only a weak connection with the coding performance measures, except for the synthetic images.

8.7.2. Hybrid Wavelet-based Compression System

The hybrid wavelet-based compression system (HWCS) described in chapter 7 demonstrated that it is possible to significantly improve the wavelet encoder output and yet be able to maintain a reasonable visual image quality. In some preliminary work described in section 7.14 of this thesis has shown that the HWCS can be incorporated into a complete wavelet-based compression system for mobile telephones.

One possible direction for further work is an automatic cropping algorithm to replace the present cropping procedure. Presently, the subject of interest is manually defined by the user prior to being cropped. One suggestion is to explore the utilization of existing object recognition techniques to implement the automatic cropping algorithm. An artificial intelligence (AI) system is also a possibility.

One drawback of the averaging algorithm used in the HWCS is that it suffers from similar blocky artefacts to those produced by the JPEG-based algorithm. Therefore, one possible avenue for future work is to eliminate the blocky artefacts produced by the averaging algorithm and thereby improve the visual quality of the reconstructed images. At this stage it is conjecture that these blocky artefacts are caused by the negligence of correlations among adjacent blocks and by a coarse averaging of pixels independent of these blocks. A possible solution to this problem is to adopt the Yung et al [1996] approach discussed in section 3.5. The Yung et al [1996] approach was to characterize and quantify the blocking artefact and then propose an iterative post-processing approach to remove it.

Finally, another possible direction for future work is to consider a hardware implementation of the wavelet encoder and decoder used in the proposed wavelet-based compression system for mobile telephones discussed in section 7.14. A special-purpose very large scale integrated (VLSI) chip may be designed and fabricated to implement the wavelet encoder and decoder. This could significantly increase the computational speed of the wavelet encoder and decoder and established its suitability for mobile applications.

8.7.3. Human Vision Perception and Image Evaluation

It would be interesting to investigate further why most of the respondent in the image evaluation reported in section 7.13 prefer the larger block size images. A detailed study of human vision perception might reveal the reasons why the respondents chose the larger block sizes. The study should include other forms of image evaluation methods based on weighted mean squared errors (WMSE) [Taubman et al, 2002, pp.199-207]. WMSE is based on the human visual system contrast sensitivity function (CSF). The reason for including WMSE in the image evaluation is that it is a more accurate gauge of perceptual quality than the more usual mean squared error (MSE) [Taubman et al, 2002, pp.621].

In conclusion, this thesis has demonstrated that a hybrid approach to image compression, taking account of display device capabilities, can offer significant improvements in image compression for mobile applications.

References

Adams and Ward, 2001	Adams, M, Ward, R., "Wavelet Transforms in the JPEG-2000 Standard", *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, BC, Canada, Vol.1, pp.160-163, 2001
Adams, 2002	Adams, M, "The JPEG-200 Still Image Compression Standard", ISO/IEC TC 1/SC 29/WG 1, N 2412, December, 2002
Ahumada, 1995	Ahumada, A. J., "A Fast DCT Block Smoothing Algorithm", *Visual Communication and Image Processing Annual Meeting Proceedings*, SPIE Vol. 2501, Paper 5, 1995.
Antonini et al, 1992	Antonini, M, Barlaud, M, Mathieu, P, Daubechies, I, "Image Coding Using Wavelet Transform", *IEEE Transactions on Image Processing*, Vo.1, No.2, pp.205-220, April, 1992
Atsumi and Farvardin, 1998	Atsumi, E., and Farvardin, N., "Lossly/Lossless Region-Of-Interest Image coding Based On Set Partitioning In Hierarchical Tress", *Proceedings of IEEE International Conference in Image Processing*, pp.87-91, October 1998
Augusteijn et al, 1995	Augusteijn, M.F., Clemens, L.E., and Shaw, K.A., "Performance Evaluation of Texture Measures for Ground Cover Identification in Satellite Images by Means of a Neural Network Classifier", *IEEE Transactions Geoscience and Remote Sensing*, Vol. 33, pp.616 – 626, May 1995
Balmer, 1998	Balmer, L. (1998). **Signals and Systems – An Introduction**. Hertfordshire: Prentice Hall
Bolton, 1995	Bolton, W. (1995). **Fourier Series**. Essex, England: Longman Scientific and Technical.
Burdick, 2000	Burdick, H.E. (2000), **Digital Imaging – Theory and Applications**. Singapore, McGraw-Hill
Burrus et al, 1998	Burrus, C.S, Gopinath, R.A, Guo, H. (1998), **Introduction to Wavelets and Wavelets Transforms – A Primer**. New Jersey: Prentice Hall.
Burt and Adelson, 1983	Burt, P.J, Adelson, E.H, "The Laplacian Pyramid as a Compact Image Code", *IEEE Transactions on Communications*, com-31, No. 4, pp.532-540, 1983.
Carnathan, 2004	Carnathan, B, "Backgrounds", Online. Google. http://www.photo.net/learn/backgrounds/ View date: July 2004

Cave, 2002 — Cave, M, "Review of Radio Spectrum Management", *Department of Trade and Industry, and Her Majesty's Treasury*, March 2002.

Christopoulos et al, 2000a — Christopoulous, C., Skodras, A., Ebrahimi, T., " The JPEG2000 Still mage Coding System: An Overview", *IEEE Transactions on Consumer Electronics*, Vo.46, No.4, pp.1103-1127, November 2000

Christopoulos et al, 2000b — Cristopoulos, C., Askelof, J., and Larsson, M., "Efficient ethods for Encoding Regions of Interest in The Upcoming JPEG2000 Still Image Coding Standard", *IEEE Signal Processing Letters*, September 2000.

Cosman et al, 1996 — Cosman, P.C, Gray, R.M, Vetterli, M, "Vector Quantization of Image Subbands: A Survey", *IEEE Transactions on Image Processing*, Vol.5, pp. 202-225, February 1996

Crochiere et al, 1976 — Crochiere, R.E, Webber, Flangan, J.L, "Digital coding of Speech in Sub-bands", *Bell System Technical Journal*, Vol.55, pp. 1069-1086, 1976.

Croisier et al, 1976 — Croisier A, Esteban D, Galand C, "Perfect Channel Splitting by use of Interpolation/Decimation/Tree Decomposition Techniques", *International Conference on Information Sciences and Systems*, pp.443 – 446, Patras, Greece, 1976

Cruz et al, 1999 — Cruz, D.S., Ebrahimi, T., Larsson, M., Askelof, J., Cristopoulos, C., "Region of Interest Coding in JPEG2000 for Interactive Client/Server Applications", *Proceedings of the IEEE Third Workshop on Multimedia Signal Processing (MMSP)*, pp. 389-394, September 1999

Daubechies, 1988 — Daubechies, I, "Orthogonal Bases of Compactly Supported Wavelets", *Communications on Pure and Applied Mathematics*, pp.909 – 996, 1988

Daubechies, 1992 — Daubechies, I, (1992), **Ten Lectures on Wavelets**, SIAM

DeVore et al, 1992 — DeVore, R.A., Jawerth, B., Lucier, B.J., "Image Compression through Wavelet Transform Coding," *IEEE Transactions on Information Theory*, Vol.32, No.2, pp.719-746, March 1992

Dulyakarn et al, 2000 — Dulyakarn, P, Rangsanseri, Y, Thitimajshima, P, "Comparison of Two Texture Features For Multispectral Imagery Analysis", *Asian Conference on Remote Sensing*, 2000.

Efford, 2000 — Efford, N, (2000), **Digital Image Processing – a practical introduction using Java**, Addison Wesley.

Eskicioglu and Fisher, 1995 — Eskicioglu, M, Fisher, P. S. "Image Quality Measures and Their Performance," *IEEE Transactions on Communications*, Vol. 43, No. 12, December 1995, pp. 2959-2965

Frendendall and Behrend, 1960 — Frendendall, G.L, Behrend, W.L, "Picture Quality – Procedures for Evaluating Subjective Effects of Interference", *Proceedings IRE*, 48, pp.1030 – 1034, 1960

Gabor, 1946 — Gabor, D, "Theory of Communication", *Journal of the IEE*, 93, pp.429-457, 1946

Gevers et al, 2000 — Gevers, T, Aldershoff, F, Smeulders A.W.M, "Classification of Images on Internet by Visual and Textual Information", *Proceedings of SPIE*, Vol. 3964, 2000

Gharavi and Tabatabai, 1988 — Gharavi, H, Tabatabai, A, "Sub-band Coding of Monochrome and Colour Images", *IEEE Transactions on Circuits and Systems*," Vol.35, February, 1988

Gonzales and Woods, 1992 — Gonzales, R and Woods, R, **Digital Image Processing**, Addison-Wesley Publishing Company, 1992.

Goswami and Chan, 1999 — Goswami, J, Chan A.K. (1999). **Fundamentals of Wavelets – Theory, Algorithms and Applications**. John Wiley

Grigorescu et al, 2003 — Grigorescu, S.E , Petkov, N, "Texture Analysis Using Renyi's Generalised Entropies", *International Conference on Image Processing 2003(ICIP 2003)*, Vol.1, pp.1-241-4, September 2003

Howard and Vitter, 1992 — Howard and Vitter, "Practical Implementation of Arithmetic Coding", *Brown University, Department of Computer Science*, TR-92--18, revised version, April 1992.

Hubbard, 1998 — Hubbard, B.B,(1998), **The World According to Wavelets**, AK Peters

Huffman, 1952 — Huffman, D.A., "A Method for Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*, Vol.40, pp.1098-1101, September 1952

Huh and Hwang, 1997 — Huh, Y. and Hwang, J. J., "The New Extended JPEG Coder with Variable Quantizer Using Block Wavelet Transform" *IEEE Transactions on Consumer Electronics*, 43, No.3, pp. 401-409, 1997

ISO, 1992 — ISO/IEC 10918-3 ITU_T Recommendation T.84, "Information technology – Digital Compression and Coding of Continuous-tone Still Images: Requirements and Guidelines", 1992

Iyer and Bell, 2001 — Iyer, L.R., Bell, A.E., "Improving Image Compression Performance with Balanced Multiwavelets," *Asilomar Conference on Signals, Systems and Compression*, Vol.1, pp.773-777, November 2001

Jbig, 2002 — JBIG Offical home page. Online. Google. http://www.jpeg.org/public/jbighomepage.htm. View date: January 2002

Kingsbury, 2003 — Kingsbury, N, "Image Characteristics", *The Connexions Project*, 2003. Online. Google. http://cnx.rice.edu/content/m11085/latest/ View date: February 2004

Kretzmer, 1956 — Kretzmer, E.R, "Reduced-alphabet Representation of Television Signals", *IRE Convention Record*, pp.140, 1956

LeGall and Tabatabai, 1988 — LeGall, D, Tabatabai, A, "Subband Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques", *IEEE International Conference on Acoustic, Speech, Signal Processing*, NewYork, Vol.2, pp.761–764, April 1988

Lervik and Ramstad, 1995 — Lervik, J.M, Ramstad, T.A., "Subband Compression of High Quality Color Still Images", *Proceedings of NOBIM-95"*, Alessund, Norway, pp. 75-80, June 1995

Lewis and Traill, 1999 — Lewis, J.P., Traill, A., (1999) **Statistics Explained**, Addison-Wesley

Liang et al, 2003 — Liang, J, Tu, C, Tran, T.D, "Optimal Pre- and Post-Processing for JPEG2000 Tiling Artifact Removal", *Conference on Information Sciences and Systems*, The Johns Hopkins University, March 2003

Limb and Rubinstein, 1974 — Limb, J.O., Rubinstein, C.B., "Plateau Coding of the Chrominance Component of Color Picture Signals", *IEEE Transactions on Communications*, Vol. Com-27, No.6, pp.812-820, June 1974

Liu and Chan, 1998 — Liu, C.S, Chan, A.K. (1998). **Wavelet Toolware**. London: Academic Press Limited.

Liu, 2004 — Liu, N, "Mini-Project #1: Spatial Masking", Online. Google. http://people.cornell.edu/pages/nwl2/project1-webpage/project1-webpage.htm. View date: March, 2004.

Lowes, 2002 — Lowes,R, "Computer Consult: Toward a handheld EMR". *Medical Economics*, 2002

Mallat, 1989a — Mallat, S.G, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11, pp. 674-692, 1989

Mallat, 1989b — Mallat S.G, "Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(R)$", *Transactions of the American Mathematical Society*, Vol. 315, No.1, pp. 69-87, 1989

Misiti, 1997 — Misiti, M, Misiti, Y, Oppenheim, G, Poggi, J.M. (1997). **Wavelet Toolbox**. The MathWork, Inc.

Moffat et al, 1995 — Moffat, Neal, R.M, and Witten, I.H, "Arithmetic Coding Revisited", *Proceedings of Data Compression Conference (DCC '95)*, pp.202 – 211, March 1995

Nasrabadi and King, 1988 — Nasrabadi, N. M, King, R. A., "Image coding using vector quantization: A review," *IEEE Transactions on Communications.* Vol.36, pp.957-971, 1988.

Netravali and Rubinstein, 1979 — Netravali, A.N., Rubinstein, C.B., "Luminance Adaptive Coding of Chrominance Signals", *IEEE Transactions on Communications*, Vol. Com-27, No.4, pp.703-710, April 1979

Oh and Woolley, 1999 — Oh, J and Woolley, S. I., "Diagnostic Quality Test for Wavelet Compressed Digital Angiogram Images", *Proceedings of Image Processing and its Applications*, No. 465, 512- 516, 1999

Ortega and Vetterli, 1997 — Ortega, A, Vetterli, M, "Adaptive Scalar Quantization Without Side Information", *IEEE Transactions on Image Processing*, Vol.6, pp 665-676,1997

Parker, 1997 — Parker, J.R. (1997). **Algorithms for Image Processing and Computer Vision**, Wiley, USA

Phillips, 2003 — Phillips, W.J, "Time-Frequency Analysis", 2003. Online. Google. www.enmath.dal.ca/courses/engm6610/notes/node3.html View date: January 2003

Pratt, 1971 — Pratt, W.K., "Spatial Transform Coding of Color Images", *IEEE Transactions on Communication Technology*, Vol.com-19, No.6, pp.980-992, 1971

Pratt, 1991 — Pratt, W.K. (1991). **Digital Image Processing**, Wiley-Interscience, 2nd Edition, USA

Qian, 2002 — Qian, S. (2002), **Introduction to Time-Frequency and Wavelet Transforms**, Prentice Hall, Upper Saddle River, NJ 07458

Queiroz et al, 1997 — Queiroz, R., Choi, C. K., Huh Young, and Rao, K. R. "Wavelet Transform in a JEPG-Like Image Coder", *IEEE Transactions on Circuits and Systems For Video Technology*, Vol.7, pp.419 – 424, April 1997

Rao and Bopardikar, 1998 — Rao, R.M and Bopardikar, A.S. (1998). **Wavelet Transforms – Introduction to Theory and Applications**, Addison Wesley Longman, Inc.

Ratib et al, 2003 — Ratib, O, McCoy, J.M., McGill, D.R., Li, M, Brown, A, "Use of Personal Digital Assistants for Retrieval of Medical Images and Data on High-Resolution Flat Panel Displays", *Radiographics*. January, 2003

Rioul and Vetterli, 1991 — Rioul, O, Vetterli, M, "Wavelets and Signal Processing", *IEEE Signal Processing Magazine*, Vol.8, pp.14-38, October 1991

Rissanen and Langdon, 1979 — Rissanen, J and Langdon, G. G, "Arithmetic Coding", *IBM Journal of Research and Development*, 23, No.2, pp.149-162, March 1979.

Rout and Bell, 2002 — Rout, S, Bell, A.E., "Color Image Compression: Multiwavelets vs. Scalar Wavelets", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, pp. 3501-3504, Orlando, 2002.

Saenz et al, 1999 — Senz, M., Salama, P., Shen, K., Delp, E.J., "An Evaluation of Color Embedded Wavelet Image Compression Techniques," *Proceedings of the SPIE/IS&T Conference on Visual Communications and Image Processing (VCIP)*, San Jose, CA, pp. 282-293., January 23-29 1999

Saha and Vemuri, 1999a — Saha, S. and Vemuri, R., "Adaptive Wavelet Filters in Image Coders - How Important are They?". *Conference Proceedings. IEEE IECON '99*, San Jose, California, 1999

Saha and Vemuri, 2000a — Saha, S. and Vemuri R., "How Do Image Statistics Impact Lossy Coding Performance". *Proc. IEEE International Conference on Information Technology: Coding and Computing (ITCC)*, 2002.

Saha and Vemuri, 2000b — Saha, S. and Vemuri R. "An Analysis on the Effect of Image Features on Lossy Coding Performance", *IEEE Signal Processing Letters*, 2000.

Saha and Vemuri, 2000c — Saha, S, Vemuri, R, "Effect of Image Activity on Lossy and Lossless Coding Performance", *Data Compression Conference (DCC '00)*, 2000

Said and Pearlman, 1996 — Said, A, Pearlman, W.A., "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Transaction on Circuits and Systems for Video Technology*, Vol.6, No.3, June 1996.

Santa-Cruz and Ebrahimi, 2000a — Santa-Cruz, D. and Ebrahimi, T., "An Analytical Study of JPEG 2000 Functionalities", *International Conference on Image Processing*, September 2000

Santa-Cruz and Ebrahimi, 2000b — Santa-Cruz, D. and Ebrahimi, T. "A Study of Jpeg 2000 Still Image Coding versus Other Standards", *Conference Proceedings of the X European Signal Processing Conference,* September 2000

Santa-Cruz et al, 2000 — Santa-Cruz, D., Ebrahimi, T. Askelof J, Larsson M, Christopoulos C A., "JPEG2000 Still Image Coding versus Other Standards". *Conference Proceedings. SPIE Applications of Digital Image Processing XXIII.* Vol. 4115, p. 446-454, 2000

Schiller, 2000 — Schiller, J, (2000), **Mobile Communications**, Addison-Wesley, Harlow

Shapiro, 1993 — Shapiro, J.M., "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, Vol. 41, No.12, pp.3445-3462, December 1993

Shen and Delp, 1997 — Shen, K., Delp, E.J., "Color Image Compression Using an Embedded Rate Scalable Approach," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, California, pp. III-34-II-37, October 26-29 1997

Shen, 1997 — Shen, K., "A Study of Real-time and Rate Scalable Image and Video Compression," Ph.D. Thesis, School of Electrical and Computer Engineering, Purdue University, December 1997

Smith and Chang 1997 — Smith, J. R, Chang, S. F, "Multi-stage Classification of Images from Features and Related Text", *Proceedings Fourth DELOS workshop*, Pisa, Italy, August, 1997

Sprljan et al, 2003 — Sprljan, N., Grgic, S., Grgic, M., "ImAn - Educational Tool for Image Analysis," *Proceedings of the IEEE Region 8 EUROCON 2003*, Ljubljana, Slovenia, pp. 238-242, September 2003

Strang and Nguyen, 1997 — Strang, G, Nguyen, T, **Wavelets and Filter Banks**. USA: Wellesley-Cambridge Press, 1997

Taubman et al, 2002 — Taubman, D.S, Marcellin, M.W, **JPEG 2000 Image compression fundamentals, standards and practice**, Kluwer Academic Publishers, 2002

Umbaugh, 1998 — Umbaugh, S.E, (1998). **Computer Vision and Image Processing – a practical approach using CVIPtools**, Prentice Hall

Unser, 2003 — Unser, M., "Wavelet Theory Demystified", *IEEE Transactions on Signal Processing*, Vol. 51, pp. 470-483, 2003

Valens, 2004 — Valens, C., "EZW Encoding", Online. Google. http: perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html View date: June 2004

Vanhoucke, 2001 — Vanhoucke, V, "Block Artifact Cancellation in DCT Based Image Compression", Project Report, Stanford University, March 2001. Online. Google. http://www.ee.columbia.edu/~marios/courses/e6820y02/project/paper_reviews.html View date: October 2004

Vetterli and Herley, 1992 — Vetterli, M, Herley, C, "Wavelets and Filter Banks: Theory and Design", *IEEE Transactions on Signal Processing.*, 40(9), pp.2207-2232, 1992

Vetterli and Kovacevic, 1995 — Vetterli, M, Kovacevic, J. (1995). **Wavelets and Subband Coding**. Prentice Hall

Vetterli, 1984 — Vetterli, M, "Multi-dimensional Sub-band Coding: Some Theory and Algorithm", *Signal Processing*, Vol. 6, pp.97-112, 1984.

Villasenor et al, 1995 — Villasenor, J.D., Belzer, B, Liao, J., "Wavelet Filter Evaluation for Image Compression", *IEEE Transactions on Image Processing*, 1995

Wallace, 1992 — Wallace, G.K., "The JPEG Still Picture Compression Standard", *IEEE Trans. on Consumer Electronics*, 38, No. I, pp.18-34, 1992.

Watson, 1993 — Watson, A.B, (1993). **Digital Images and Human Vision**, MIT Press.

Woods and O'Neil, 1986 — Woods J.W, O'Neil S.D, "Subband Coding of Image", *IEEE Trans. Acoust, Speech, Signal Processing*, ASSP-34, No.5, October 1986

Yap and Comley, 2004 — Yap,V.V., Comley, R.A., "A Segmentation-based Wavelet Compression Scheme for Still Images", *IASTED International Conference on Signal and Image Processing (SIP2004)*, Honolulu, 2004

Yap and Rahman, 2003a — Yap,V.V., Rahman, S, "Performance of Wavelet-based Image Compression on Synthetic Images", *International Conference on Robotics, Vision, Information And Signal Processing*, IEEE Malaysia, pp. 51-56, Penang, Malaysia , January 2003

Yap and Rahman, 2003b — Yap,V.V., Rahman, S, "The Implications of Image Statistics and Image Features on Coding Performance of Synthetic Images", *The 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003)*, Orlando, Florida, July 2003

Yung et al, 1996 — Yung, K, Li, J, Kuo, J, "Image Enhancement for Low Bit-rate JPEG and MPEG Coding via PostProcessing", *SPIE: Visual Communication and Image Processing*, Vol.2727, pp. 1484-1494, Orlando, FL, 1996.

Appendix 1

This appendix contains the complete set of coding performance measures for sixty-four images.

Natural images

Image	Haar				db2				bior4.4				bior6.8				coif4				sym4			
	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp
		CCR	CR			CCR	CR			CCR	CR			CCR	CR			CCR	CR			CCR	CR	
bird	36.66	42.24	**45.18**	0.177	36.98	41.77	44.65	0.179	37.23	38.92	41.57	0.192	37.26	32.04	32.67	0.245	37.01	27.84	28.09	0.285	**37.29**	38.83	40.90	0.196
bulehills256	36.98	53.37	**47.84**	0.167	37.59	52.63	47.67	0.168	37.61	44.77	40.12	0.199	**37.75**	35.43	31.50	0.254	37.72	30.74	26.46	0.302	37.68	45.88	42.06	0.190
columns	33.76	33.22	30.58	0.262	33.96	33.23	**31.20**	0.256	33.91	32.47	29.31	0.273	33.92	26.59	23.13	0.346	33.67	22.59	19.23	0.416	**34.04**	30.25	28.14	0.284
coupla	32.49	25.77	23.43	0.341	33.02	28.10	**25.67**	0.312	33.25	28.82	24.86	0.322	**33.41**	23.61	19.70	0.406	33.31	21.07	16.68	0.480	33.35	27.44	24.33	0.329
8mila02_256	32.58	27.24	24.66	0.324	33.65	30.48	**26.47**	0.302	34.22	28.55	24.06	0.333	**34.40**	23.16	18.38	0.435	34.15	20.25	15.49	0.517	34.07	28.23	24.30	0.329
marcia	31.08	20.29	19.16	0.416	31.77	21.40	**18.78**	0.404	32.03	20.34	17.77	0.450	**32.18**	16.81	13.68	0.585	32.03	15.19	11.46	0.698	32.01	20.08	17.98	0.445
womanbaby256	33.64	33.48	31.22	0.256	34.80	37.90	**32.34**	0.247	35.25	36.42	30.20	0.265	**35.46**	29.40	23.57	0.339	35.17	25.64	20.32	0.394	35.21	35.94	30.38	0.263
justinkelly256	29.76	13.92	14.14	0.566	30.38	**15.50**	**15.10**	0.530	30.78	16.38	14.60	0.548	**30.92**	14.35	11.89	0.673	30.78	13.04	10.09	0.793	30.71	15.49	14.26	0.561
8mila01_256	30.62	19.38	18.15	0.441	31.20	20.86	**19.11**	0.419	31.53	20.87	17.68	0.453	**31.69**	17.84	14.16	0.565	31.46	16.67	12.49	0.640	31.41	20.08	17.67	0.453
sail	28.32	10.95	11.25	0.711	28.58	11.70	**11.91**	0.672	28.79	12.07	11.42	0.700	**28.98**	10.66	9.25	0.865	28.80	9.77	8.02	0.997	28.80	11.53	11.11	0.720
aerial02_256	28.20	8.33	9.45	0.847	28.26	8.76	**9.75**	0.821	28.27	9.38	9.57	0.836	**28.41**	8.19	7.56	1.058	28.18	7.46	6.52	1.227	28.33	8.96	9.37	0.853
aerial01_256	29.79	14.93	16.09	0.497	29.98	15.81	**16.22**	0.493	29.86	16.01	15.67	0.511	**30.07**	13.55	12.03	0.665	29.87	12.33	10.44	0.766	29.94	15.69	15.79	0.507
dallastwo256	29.95	12.80	12.76	0.627	29.82	13.15	**13.29**	0.602	**30.01**	13.84	12.59	0.635	29.99	12.84	10.92	0.733	29.85	12.07	9.64	0.830	30.00	13.23	12.48	0.641
peppers	30.28	14.17	13.40	0.597	30.90	15.28	**14.57**	0.549	31.29	14.95	12.98	0.616	**31.32**	12.04	9.92	0.807	31.23	10.54	8.16	0.980	31.23	14.30	13.06	0.613
balloons	34.24	25.70	23.26	0.344	34.32	27.73	**25.58**	0.313	34.51	27.93	24.85	0.322	**34.61**	22.99	19.57	0.409	34.42	20.71	17.31	0.462	34.52	26.11	23.66	0.338
mountain256	28.05	8.73	10.15	0.788	28.14	9.09	**10.35**	0.773	28.19	9.34	9.67	0.828	**28.24**	8.71	8.17	0.980	28.15	8.19	7.12	1.123	28.16	9.03	9.64	0.830
winter256	27.47	11.39	**12.41**	0.644	27.60	11.62	12.22	0.655	27.60	11.25	11.20	0.714	**27.73**	10.44	9.42	0.850	27.65	10.02	8.43	0.949	27.68	11.19	11.21	0.714
lanacolor	30.04	16.19	16.08	0.497	30.45	17.96	**17.54**	0.456	30.64	18.78	17.08	0.468	30.75	16.15	13.60	0.588	30.59	14.52	11.58	0.692	30.61	17.72	16.62	0.481
palmbladas	26.67	5.96	6.72	1.190	26.96	6.62	**7.25**	1.103	27.27	7.17	7.21	1.109	**27.40**	6.62	6.06	1.320	27.25	6.28	5.34	1.498	27.19	6.89	7.10	1.127
boy	30.33	17.81	18.32	0.437	30.91	19.81	**19.43**	0.412	31.10	18.97	17.67	0.453	**31.22**	15.78	13.43	0.596	30.97	14.19	11.51	0.695	31.07	18.97	17.93	0.446
flower256	36.70	51.63	58.78	0.14	**37.315**	49.16	58.86	0.141	37.22	42.67	47.25	0.169	37.239	34.99	38.573	0.207	37.124	30.18	31.98	0.250	37.175	43.20	50.92	0.157
aerial03b_256	**26.04**	5.43	6.22	1.29	25.934	5.406	8.18	1.295	25.955	5.533	5.805	1.378	25.90	5.211	4.962	1.612	25.862	5.006	4.382	1.826	25.863	5.43	5.873	1.362

Synthetic images

Image	Haar				db2				bior4.4				bior6.8				coif4				sym4			
	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp
		CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR	
snowwhite02	32.08	22.47	21.64	0.370	32.64	24.65	22.70	0.352	32.70	25.48	21.97	0.364	33.14	21.16	16.99	0.471	32.73	18.64	14.47	0.553	32.72	24.17	21.44	0.373
toystory02_256	31.35	16.73	15.74	0.508	31.55	16.87	16.24	0.493	31.69	17.23	15.54	0.515	31.77	15.11	12.79	0.626	31.57	13.62	11.02	0.726	31.54	16.64	15.35	0.521
toystory01_256	30.59	14.20	14.20	0.563	30.49	14.83	14.97	0.534	30.68	14.81	13.97	0.573	30.69	13.16	11.44	0.699	30.51	12.27	9.88	0.809	30.58	14.73	14.14	0.809
mermaid01_256	29.74	9.69	9.39	0.852	29.67	9.78	9.83	0.814	29.81	9.93	9.33	0.858	29.82	8.71	7.57	1.057	29.69	7.82	6.40	1.250	29.71	9.60	9.30	0.860
ebugslife02_256	29.81	12.92	13.20	0.606	30.15	13.02	12.98	0.616	30.32	13.10	12.03	0.665	30.48	11.29	9.59	0.834	30.42	10.15	8.05	0.994	30.38	12.77	11.96	0.669
bugsbunny02_256	30.87	13.26	12.80	0.625	30.70	13.55	13.42	0.596	30.75	14.26	13.51	0.592	30.81	12.96	11.50	0.696	30.60	11.66	9.74	0.821	30.70	13.63	13.17	0.608
bugsbunny01_256	30.74	14.44	13.71	0.583	30.65	14.17	13.98	0.573	30.61	14.86	13.55	0.590	30.65	12.76	10.80	0.740	30.36	11.21	8.92	0.897	30.54	14.17	13.34	0.600
geri02	30.23	14.68	14.66	0.546	30.93	16.71	16.21	0.494	31.27	17.40	15.34	0.494	31.48	14.14	11.84	0.676	31.36	12.22	9.65	0.829	31.23	16.30	15.04	0.532
gari01	29.18	10.67	10.90	0.734	30.06	12.80	12.96	0.617	30.70	14.05	13.20	0.608	30.95	11.57	10.27	0.779	30.69	10.21	8.62	0.928	30.63	13.16	12.84	0.623
mermaid02_256	31.41	11.65	11.09	0.721	31.07	11.30	11.23	0.712	30.90	11.64	10.87	0.736	30.85	10.43	9.01	0.888	30.66	9.64	7.86	1.017	30.85	11.27	10.77	0.743
hercules01_256	30.63	12.38	12.09	0.662	30.60	12.77	12.76	0.627	30.76	13.60	12.68	0.631	30.72	11.94	10.31	0.776	30.51	11.08	8.98	0.891	30.73	13.12	12.47	0.641
hercules02_256	30.22	11.53	11.23	0.713	30.29	11.72	11.67	0.686	30.31	12.56	11.63	0.688	30.35	11.64	9.99	0.801	30.23	10.98	8.76	0.913	30.34	12.06	11.43	0.700
lionking01_256	31.77	18.01	16.83	0.475	32.25	19.15	17.21	0.465	32.60	19.08	16.39	0.488	32.75	16.03	12.89	0.620	32.56	14.47	11.00	0.727	32.44	18.18	16.00	0.500
lionking02_256	33.19	27.45	24.67	0.324	34.28	28.36	24.45	0.327	34.72	26.44	22.06	0.363	35.00	21.38	16.80	0.476	34.79	19.06	14.21	0.563	34.69	25.97	21.91	0.365
monsterinc01_256	30.50	17.36	16.95	0.472	31.06	18.78	17.46	0.458	31.22	18.90	16.54	0.484	31.33	16.25	13.28	0.602	31.14	14.97	11.43	0.700	31.20	18.23	16.38	0.488
monsterinc02_256	29.80	16.85	15.85	0.505	30.25	17.74	16.73	0.478	30.39	18.53	16.03	0.499	30.38	16.07	12.81	0.624	30.27	14.64	10.97	0.729	30.38	17.98	15.91	0.503
monsterinc03_256	30.35	17.44	17.44	0.459	30.51	18.04	17.55	0.456	30.64	16.93	15.45	0.518	30.67	14.94	12.46	0.642	30.62	13.58	10.68	0.749	30.62	16.82	15.65	0.511
entz01	30.99	17.11	16.57	0.483	31.52	18.22	16.90	0.473	31.73	18.44	16.11	0.497	31.89	15.56	12.85	0.622	32.04	13.99	10.75	0.744	31.83	17.53	15.65	0.511
entz02	31.89	20.33	19.11	0.419	32.70	21.87	19.87	0.403	33.03	22.06	18.72	0.427	33.25	17.96	14.46	0.553	33.08	15.11	11.66	0.686	33.05	20.95	18.17	0.440
snowwhite01_256	32.98	28.31	25.28	0.316	33.73	32.07	27.80	0.288	33.96	32.88	26.96	0.297	34.34	27.82	21.68	0.369	34.25	24.80	18.34	0.436	34.03	31.03	26.38	0.303
windowsxp	35.93	35.48	34.09	0.235	37.175	37.27	38.10	0.222	37.639	34.97	33.55	0.238	37.733	29.38	28.27	0.283	37.531	25.59	23.45	0.341	37.64	34.45	33.92	0.236
berboon	25.87	6.47	7.26	1.102	25.968	6.64	7.257	1.102	25.969	6.70	6.765	1.183	28.02	6.323	5.847	1.368	26.01	6.12	5.24	1.528	26.00	6.66	6.89	1.162

Clipart images

Image	Haar				db2				bior4.4				bior6.8				coif4				sym4			
	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp	PSNR	Comp. Ratio		bpp
		CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR			CCR	FCR	
clipart09_256	37.13	15.76	16.68	0.480	33.02	9.92	11.27	0.710	31.80	9.78	10.64	0.738	30.63	7.82	7.62	1.050	30.54	6.93	8.30	1.271	30.98	9.83	10.63	0.752
clipart06_256	41.34	51.21	43.28	0.185	36.07	33.70	32.58	0.246	35.81	30.68	29.21	0.274	35.38	26.67	24.13	0.332	34.54	23.64	20.17	0.397	35.78	30.27	29.44	0.272
clipart10_256	37.18	27.99	24.21	0.330	34.10	19.61	18.79	0.426	33.65	19.06	17.48	0.458	32.88	18.14	15.59	0.513	32.79	17.06	13.54	0.591	32.99	18.79	17.61	0.454
clipart05_256	35.84	19.97	16.18	0.494	32.65	15.97	14.26	0.561	32.17	16.70	13.74	0.582	31.77	14.31	11.30	0.708	31.41	13.40	9.92	0.806	32.07	16.33	13.81	0.579
clipart07_256	32.55	10.99	10.49	0.763	30.13	9.10	9.35	0.856	29.80	9.38	9.08	0.881	29.45	9.14	8.21	0.975	29.26	8.96	7.54	1.062	29.65	9.20	9.04	0.885
clipart08_256	32.56	15.51	14.79	0.541	31.56	15.73	15.44	0.518	31.65	17.36	15.96	0.501	31.44	16.16	13.70	0.584	31.27	15.68	12.51	0.639	31.38	16.43	15.63	0.512
clipart01_256	35.60	32.01	28.05	0.285	35.07	34.30	29.81	0.268	34.84	33.09	27.15	0.295	34.64	27.94	21.69	0.369	34.61	22.97	16.44	0.487	34.72	31.37	26.42	0.303
clipart03_256	34.25	21.38	19.11	0.419	32.64	18.14	17.92	0.448	32.80	16.33	16.05	0.499	32.47	12.41	11.56	0.692	32.24	12.10	10.01	0.800	32.83	16.56	16.26	0.492
clipart02_256	36.30	26.93	22.12	0.362	34.82	25.20	22.34	0.358	34.51	26.32	21.96	0.364	34.54	18.93	15.41	0.519	34.07	17.98	13.39	0.597	34.65	24.66	21.11	0.379
clipart04_256	32.28	8.49	8.35	0.958	29.16	6.17	6.37	1.257	28.77	6.61	6.34	1.261	28.21	6.51	5.84	1.370	28.17	6.27	5.20	1.539	28.56	6.34	6.25	1.280
clipart11_256	37.75	25.76	23.78	0.337	33.89	18.04	18.41	0.435	33.42	17.82	17.18	0.466	32.35	15.95	14.71	0.544	32.36	14.69	12.33	0.649	33.03	17.37	17.25	0.464
clipart12_256	35.90	12.90	11.58	0.691	31.10	8.88	8.93	0.896	30.15	8.46	8.19	0.977	29.07	7.00	6.21	1.288	28.92	6.03	4.99	1.605	29.79	8.28	8.02	0.997
clipart13_256	45.03	63.55	57.64	0.138	38.31	54.09	50.32	0.159	39.09	54.75	50.35	0.159	37.60	51.09	43.81	0.183	38.11	47.96	40.88	0.196	37.56	51.88	48.62	0.165
clipart14_256	38.56	26.72	40.00	0.200	38.78	19.91	30.72	0.260	35.80	17.24	23.48	0.341	34.93	11.26	12.27	0.652	34.39	9.95	9.77	0.819	34.37	16.30	22.45	0.356
clipart15_256	39.64	23.77	37.87	0.211	34.98	16.49	23.37	0.342	34.06	14.21	18.52	0.432	32.99	10.21	10.86	0.737	32.39	8.98	8.65	0.925	32.79	13.91	18.19	0.440
clipart16_256	39.93	25.24	35.88	0.224	36.22	19.18	25.36	0.315	36.51	17.36	21.54	0.371	35.33	11.04	12.73	0.628	34.57	9.75	10.23	0.782	34.82	16.64	21.45	0.373
clipart17_256	34.77	14.91	12.35	0.648	30.73	9.61	8.95	0.894	30.07	9.25	8.40	0.953	29.41	8.56	7.37	1.085	29.14	8.00	6.42	1.246	29.62	9.08	8.44	0.948
clipart18_256	34.17	16.91	14.76	0.542	31.32	15.11	14.38	0.556	31.06	16.37	14.58	0.549	30.79	15.64	13.27	0.603	30.57	15.34	12.06	0.663	30.90	15.45	14.20	0.563
clipart19_256	33.88	14.67	13.68	0.586	31.18	12.21	12.18	0.657	31.09	12.94	12.27	0.652	30.61	13.00	11.36	0.704	30.28	12.57	10.24	0.781	30.54	12.45	12.07	0.663
clipart20_256	40.27	29.53	32.52	0.248	36.19	21.45	24.26	0.330	35.59	18.89	20.96	0.382	34.60	12.88	12.85	0.622	33.89	11.44	10.35	0.773	35.16	18.19	20.42	0.392

Appendix 2

This appendix contains the complete set of coding performance measures for sixty images.

RGB Image Statistics

Natural Images:

Image	R mean	G mean	B mean	R Std. Dev	G Std. Dev	B Std. Dev	R var.	G var.	B var.	R skew	G skew	B skew	R kurt	G kurt	B kurt
bird	108.17	115.48	127.46	18.77	19.02	22.00	352.17	361.94	483.92	2.80	0.11	-2.37	23.84	14.82	7.69
bulehills256	59.54	132.63	209.59	72.40	78.92	44.17	5241.80	6228.12	1951.36	0.79	0.24	-0.34	-0.93	-1.64	-1.47
columns	96.71	88.54	94.61	62.61	63.40	70.45	3919.84	4019.35	4963.07	0.64	0.60	0.88	-0.49	-0.68	-0.33
couple	100.61	76.45	73.40	51.54	53.64	65.68	2656.59	2877.36	4313.26	0.58	0.83	1.45	0.11	-0.01	1.11
8mile02_256	120.72	101.42	88.01	65.23	62.15	56.50	4255.57	3862.24	3192.32	0.07	0.53	0.65	-0.77	-0.54	-0.07
marcie	118.43	96.17	78.24	51.07	44.25	43.57	2608.10	1958.13	1898.77	0.10	0.46	0.66	-0.06	0.71	0.97
womanbaby256	242.89	211.14	196.93	22.41	43.62	56.89	502.06	1902.38	3236.52	-3.15	-0.89	-0.67	11.72	0.02	-0.78
justinkelly256	143.71	106.52	86.15	81.71	82.53	71.85	6676.23	6811.29	5162.16	-0.09	0.20	0.38	-1.41	-1.50	-1.31
8mile01_256	109.43	87.79	87.43	62.20	54.13	45.64	3868.54	2929.51	2082.97	0.25	0.31	0.41	-1.20	-1.14	-0.57
sail	97.93	112.69	133.25	43.81	35.62	37.71	1919.02	1268.95	1422.20	1.05	0.85	0.43	0.25	0.21	-0.55
aerial02_256	114.90	119.35	113.75	63.55	61.03	60.22	4038.96	3724.16	3626.78	-0.03	0.09	0.35	-1.26	-1.03	-0.89
aerial01_256	120.77	120.26	101.78	58.20	48.31	40.44	3386.86	2333.81	1635.25	0.18	-0.39	0.20	-0.81	-0.27	0.32
dallastwo256	81.05	75.16	54.51	57.22	56.95	39.10	3273.94	3242.89	1528.84	1.15	1.01	1.82	0.67	0.04	3.76
peppers	144.20	112.89	64.17	45.98	73.07	44.95	2113.97	5338.65	2020.73	-0.70	-0.11	0.84	0.00	-1.54	0.72
balloons	107.62	121.22	138.57	42.05	50.51	65.12	1768.43	2551.32	4241.19	0.18	-0.85	-0.47	-0.05	-0.42	-1.05
mountain256	141.61	133.95	141.55	65.59	68.95	78.50	4301.77	4753.86	6161.94	0.17	0.36	0.25	-1.04	-1.07	-1.49
winter256	32.74	67.46	126.73	46.47	43.08	44.91	2159.86	1855.72	2016.67	1.95	1.63	0.33	3.13	2.43	0.02
lenacolor	180.17	98.96	105.33	49.09	52.89	34.05	2410.06	2797.19	1159.66	-0.70	0.23	0.23	-0.79	-0.76	-0.17
palmblades	99.23	110.34	10.68	48.73	48.74	29.10	2374.34	2375.54	847.05	0.70	0.50	5.04	0.22	-0.14	30.29
boy	102.55	108.82	90.47	67.31	66.13	62.69	4530.52	4372.75	3930.61	0.69	0.88	1.01	-0.98	-0.39	0.19

Synthetic Images:

Image	R mean	G mean	B mean	R Std. Dev	G Std. Dev	B Std. Dev	R var.	G var.	B var.	R ekew	G ekew	B skew	R kurt	G kurt	B kurt
snowwhite02	83.65	79.07	74.57	43.15	40.87	35.65	1861.82	1670.58	1271.24	0.903	0.907	0.888	0.588	0.556	1.202
toystory02_256	165.37	151.35	117.14	81.04	85.87	68.87	6567.99	7373.49	4743.63	-0.358	-0.145	0.225	-1.531	-1.748	-1.717
toystory01_256	78.87	84.85	80.52	58.05	57.81	57.27	3369.66	3342.08	3280.07	0.842	0.684	0.749	0.035	-0.169	0.180
mermaid01_256	108.14	125.12	99.91	84.15	61.43	54.90	7080.81	3773.99	3014.52	0.222	0.507	0.508	-1.248	-0.260	-0.121
abugslife02_256	52.96	55.50	27.90	52.62	37.57	31.55	2768.99	1411.70	995.21	1.278	1.143	2.294	1.265	1.399	7.895
bugsbunny02_256	172.58	160.78	117.10	57.24	70.95	77.46	3275.87	5034.06	5999.38	-0.859	-0.833	0.291	0.116	-0.534	-1.370
bugsbunny01_256	132.10	104.30	96.12	81.15	70.19	58.47	6585.29	4927.31	3418.77	-0.005	0.368	0.739	-1.522	-1.148	-0.266
geri02	141.02	102.64	85.04	71.65	63.79	54.22	5133.07	4068.72	2939.76	-0.091	0.544	0.809	-0.827	-0.430	0.263
geri01	157.09	107.10	91.04	77.50	79.29	85.33	6006.88	6286.36	7281.68	-0.531	0.366	0.702	-0.703	-1.198	-0.950
mermaid02_256	140.63	167.81	164.93	70.34	73.20	93.07	4947.75	5358.21	8662.33	-0.443	-0.928	-0.642	-0.669	-0.670	-1.219
hercules01_256	119.66	142.37	133.03	77.38	62.48	96.68	5988.35	3903.43	9347.01	0.336	-0.183	-0.312	-1.244	-0.642	-1.688
hercules02_256	145.94	151.61	139.39	70.57	58.91	84.93	4979.71	3470.62	7212.76	-0.463	-0.317	-0.034	-0.929	-0.389	-1.291
lionking01_256	108.00	97.10	84.31	43.90	63.49	80.98	1927.54	4030.80	6557.20	-0.060	0.378	0.604	-0.304	-1.511	-1.514
lionking02_256	98.56	101.05	97.20	31.23	41.75	68.81	975.50	1742.98	4735.09	0.398	-0.267	0.124	0.930	-0.867	-1.755
monsterinc01_256	163.31	165.91	178.85	62.07	55.89	49.82	3852.79	3123.56	2482.00	0.011	-0.026	-0.644	-1.234	-1.008	-0.016
monsterinc02_256	156.43	155.14	159.46	63.28	57.61	53.93	4004.30	3318.79	2908.01	0.130	0.239	-0.252	-1.059	-0.884	-0.408
monsterinc03_256	62.63	99.78	129.93	39.60	54.26	54.14	1568.51	2944.09	2931.56	1.162	0.655	0.019	0.784	-0.738	-0.696
antz01	101.23	57.03	41.12	43.19	31.69	23.98	1865.37	1003.94	575.24	0.746	0.855	0.975	0.807	1.194	2.094
entz02	153.91	104.87	100.57	59.69	57.36	63.51	3562.98	3290.37	4033.96	-0.268	0.081	-0.311	-0.541	-0.796	-1.512
snowwhite01	95.00	90.65	84.49	52.63	49.96	38.99	2770.42	2495.72	1520.03	0.878	0.917	0.547	-0.036	-0.023	-0.530

Clipart Images:

Imege	R mean	G mean	B mean	R Std. Dev	G Std. Dev	B Std. Dev	R var.	G var.	B var.	R skew	G skew	B skew	R kurt	G kurt	B kurt
clipart09_256	188.72	105.24	154.94	111.84	125.54	124.51	12508.45	15761.14	15503.57	-1.095	0.355	-0.441	-0.802	-1.874	-1.806
clipart06_256	196.88	197.35	197.46	104.57	104.23	104.18	10935.57	10864.24	10854.24	-1.310	-1.322	-1.325	-0.222	-0.189	-0.180
clipart10_256	191.84	181.45	179.19	76.96	79.64	80.42	5923.45	6342.21	6467.92	-1.484	-0.988	-0.886	1.493	0.321	0.094
clipart05_256	234.46	203.51	178.70	40.94	58.68	79.93	1676.17	3443.89	6388.66	-4.629	-1.091	-0.562	23.748	1.363	-0.837
clipart07_256	146.19	158.13	172.53	94.35	92.19	94.18	8902.68	8498.49	8870.29	-0.220	-0.442	-0.697	-1.261	-1.071	-0.885
clipart08_256	132.37	155.94	134.29	125.65	107.10	125.43	15788.16	11470.54	15731.81	-0.072	-0.307	-0.102	-1.977	-1.676	-1.970
clipart01_256	51.09	26.21	67.49	35.46	35.31	60.49	1257.74	1246.69	3658.58	0.015	2.441	0.708	-0.821	6.791	-0.544
clipart03_256	56.66	39.05	106.62	54.25	55.58	96.99	2943.49	3088.98	9407.52	0.656	1.510	0.366	-0.935	0.682	-1.386
clipart02_256	128.52	29.22	20.56	95.21	28.97	18.57	9064.72	839.46	344.74	-0.071	2.360	6.171	-1.471	10.604	73.971
clipart04_256	198.13	186.10	119.75	96.45	90.64	114.06	9302.19	8216.08	13009.14	-1.438	-1.409	0.132	0.314	0.344	-1.861
clipart11_256	186.85	188.25	187.77	104.52	103.68	103.78	10924.83	10749.31	10770.94	-1.073	-1.107	-1.092	-0.668	-0.590	-0.617
clipart12_256	181.28	107.27	173.30	115.60	125.89	118.99	13364.03	15847.11	14159.10	-0.930	0.321	-0.770	-1.134	-1.897	-1.408
clipart13_256	255.00	219.62	219.62	0.00	88.15	88.15	0.00	7769.70	7769.70	0.00	-2.090	-2.090	0.00	2.369	2.369
clipart14_256	165.02	79.84	112.44	121.85	118.26	104.92	14848.37	13984.55	11008.54	-0.616	0.806	0.337	-1.621	-1.350	-1.453
clipart15_256	115.05	72.43	204.72	126.89	114.99	101.46	16101.46	13223.41	10293.38	0.196	0.958	-1.522	-1.962	-1.083	0.317
clipart16_256	247.34	34.78	15.70	43.52	87.52	61.30	1894.05	7659.66	3757.97	-5.507	2.119	3.647	28.331	2.489	11.303
clipart17_256	142.07	142.07	73.96	121.06	121.06	115.72	14654.53	14654.53	13390.34	-0.289	-0.289	0.925	-1.860	-1.860	-1.144
clipart18_256	193.63	223.85	200.09	78.40	54.12	65.32	6146.88	2928.57	4267.31	-1.186	-2.238	-1.221	0.382	5.055	0.845
clipart19_256	178.53	159.26	149.82	103.71	116.85	122.10	10755.32	13653.16	14909.09	-0.837	-0.509	-0.367	-1.028	-1.644	-1.809
clipart20_256	247.34	138.59	138.59	43.52	127.02	127.02	1894.05	16133.44	16133.44	-5.507	-0.175	-0.175	28.331	-1.970	-1.970

RGB Entropy Measures

Natural Images:

	Red			Green			Blue		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
bird	7.8580	4.9019	4.0235	7.8455	4.8410	3.9642	7.8202	4.9375	4.3459
bulehills256	7.8704	6.0232	4.6470	7.8642	7.2334	5.4681	7.0768	5.6306	4.4149
columns	7.8580	7.4940	5.8346	7.7879	7.4453	5.7333	7.9715	7.5467	5.9382
couple	7.9887	7.4130	5.8966	7.8765	7.2762	5.7424	8.0000	7.2640	5.8755
8mile02_256	7.9600	7.7595	6.2429	7.9425	7.6453	6.1773	7.9773	7.5377	6.1488
marcie	8.0000	7.5822	6.2948	8.0000	7.3540	6.1962	8.0000	7.2913	6.1997
womanbaby256	7.4346	4.3122	3.4614	7.6366	6.3336	4.7235	7.8329	6.8015	4.9462
justinkelly256	8.0000	7.7455	6.6643	8.0000	7.7100	6.6544	7.9129	7.5221	6.5686
8mile01_256	7.9189	7.7267	6.6131	7.8265	7.4469	6.3988	7.9189	7.4114	6.4434
sail	7.7616	7.1410	6.6456	7.6147	6.9790	6.5230	7.6511	7.0838	6.5908
aerial02_256	8.0000	7.7663	7.0567	7.9830	7.7804	7.0668	7.9830	7.7571	7.0635
aerial01_256	8.0000	7.8057	6.6781	7.9944	7.3858	6.5589	7.9600	7.1690	6.4302
dallastwo256	7.9464	7.5007	6.4227	7.8329	7.4375	6.3873	7.9366	6.8563	6.0377
peppers	7.9307	7.3332	6.2584	7.9189	7.5617	6.3776	7.8826	7.0193	6.0441
balloons	7.8138	6.8696	5.0895	7.6366	6.9795	5.1537	7.7748	6.9671	5.3110
mountain256	7.9887	7.6010	6.9138	7.9887	7.4705	6.7706	7.9887	7.2031	6.4665
winter256	7.9542	6.1218	5.7521	7.9887	7.0127	6.6130	7.9425	7.4867	7.0457
lenacolor	7.6147	7.2498	6.1988	7.8517	7.5818	6.6122	7.4512	6.9665	6.2545
palmblades	7.9425	7.5425	7.1204	7.9542	7.5790	7.1270	7.9944	2.5552	2.1606
boy	7.8138	7.0941	6.0440	7.7347	7.0911	6.0971	7.6935	6.9851	5.9390

Synthetic Images:

	Red			Green			Blue		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
snowwhite02	7.9425	7.2809	6.1975	7.9484	7.1898	6.1461	8.0000	7.0680	6.0953
toystory02_256	7.9009	7.0150	5.6206	7.9069	6.6024	5.3025	7.8455	6.7195	5.4353
toystory01_256	8.0000	7.4877	6.3619	8.0000	7.5870	6.4021	8.0000	7.5238	6.4073
mermaid01_256	8.0000	7.4701	6.2368	8.0000	7.4496	6.2266	8.0000	7.4393	6.2979
abugslife02_256	8.0000	7.0986	6.3911	7.9368	7.0055	6.4229	7.9887	6.1608	5.7538
bugsbunny02_256	8.0000	6.1371	4.9484	8.0000	6.0586	4.7909	8.0000	6.2011	4.9738
bugsbunny01_256	8.0000	7.7175	6.5470	8.0000	7.6952	6.4751	8.0000	7.4639	6.3908
geri02	8.0000	7.5042	6.3809	8.0000	7.7971	6.6593	7.9658	7.5573	6.5252
geri01	8.0000	6.8170	5.6559	8.0000	7.1370	6.0182	8.0000	6.8271	5.8569
mermaid02_256	8.0000	7.5385	5.8928	8.0000	7.0274	5.5477	8.0000	6.4397	5.2780
hercules01_256	8.0000	7.3257	5.9840	8.0000	7.3235	5.8541	8.0000	6.8234	5.5271
hercules02_256	8.0000	7.7763	6.5573	8.0000	7.6034	6.3625	8.0000	7.3285	6.1442
lionking01_256	7.9189	7.1450	5.9561	7.8138	6.9545	5.6849	7.8202	6.4329	5.4075
lionking02_256	7.7279	6.7870	5.6004	7.6439	7.0464	5.6208	7.7748	7.0346	5.6180
monsterinc01_256	7.9830	7.5399	6.3523	7.9484	7.5287	6.3587	8.0000	7.4795	6.4060
monsterinc02_256	7.9830	7.4776	6.4485	7.9425	7.4739	6.4569	8.0000	7.6412	6.6051
monsterinc03_256	7.9830	6.8948	6.1946	7.9307	7.5077	6.5399	8.0000	7.7099	6.7093
antz01	8.0000	7.3617	6.4695	7.8265	6.9064	6.1387	7.5392	6.4395	5.8387
antz02	8.0000	7.6985	6.1749	7.9425	7.6435	5.9949	7.8392	7.0342	5.6066
snowwhite01	7.8329	7.4166	6.0827	7.7616	7.3299	6.0009	7.7074	7.1382	5.9405

Clipart Images:

	Red			Green			Blue		
Imege	H0	H1	H2	H0	H1	H2	H0	H1	H2
clipart09_256	1.0000	0.8266	0.5079	1.0000	0.8779	0.6291	1.0000	0.9663	0.6340
clipart06_256	1.5850	0.9495	0.5418	2.0000	0.9522	0.5465	1.5850	0.9454	0.5428
clipart10_256	1.5850	1.3922	0.8077	1.5850	1.3922	0.8077	1.5850	1.3922	0.8077
clipart05_256	4.3219	3.0508	1.7335	4.5236	3.2774	1.8658	4.4594	3.2455	1.8508
clipart07_256	6.8826	3.1461	1.9886	6.8948	2.9497	1.8809	6.6865	2.1739	1.4170
clipart08_256	7.9715	1.8006	1.2165	7.9773	3.4424	2.6337	7.9773	1.8588	1.2698
clipart01_256	6.7549	5.4530	3.2489	7.1997	4.7733	2.8833	7.5622	5.2958	3.1550
clipart03_256	7.2761	5.0388	3.0300	7.2946	4.7646	2.9033	7.8948	4.3134	2.6003
clipart02_256	7.6294	4.2547	2.4609	6.7004	3.1437	1.8228	5.1699	3.8294	2.2215
clipart04_256	1.5850	1.2527	0.8726	2.0000	1.9177	1.2177	2.0000	1.9177	1.2177
clipart11_256	6.5392	1.5900	1.0210	6.9658	1.6102	1.6102	6.9658	1.8109	1.0447
clipart12_256	1.0000	0.8675	0.5381	1.0000	0.9818	0.6369	1.0000	0.9048	0.5752
clipart13_256	0.0000	0.0000	0.0000	1.0000	0.5809	0.3093	1.0000	0.5809	0.3093
clipart14_256	1.0000	0.9366	0.5561	1.0000	0.8967	0.5269	1.5850	1.5818	0.8850
clipart15_256	1.0000	0.9931	0.5739	1.0000	0.8809	0.5263	1.0000	0.7182	0.4552
clipart16_256	1.0000	0.1945	0.1566	1.0000	0.5747	0.3403	1.0000	0.3337	0.1942
clipart17_256	1.5850	1.4326	0.8824	1.5850	1.4326	0.8824	1.0000	0.8888	0.5364
clipart18_256	2.5850	1.9540	1.1825	2.3219	1.3696	0.8579	2.5850	2.0269	1.2235
clipart19_256	2.0000	1.5465	0.9515	2.3219	1.5462	0.9566	1.5850	1.2288	0.7150
clipart20_256	1.0000	0.1845	0.1566	1.0000	0.9945	0.5841	1.0000	0.9945	0.5841

Image Gradient, Spatial Frequency, and Spectral Flatness Measure Results (RGB)

Natural Images:

Image	ImgGrad (r)	ImgGrad (g)	ImgGrad (b)	SF (r)	SF (g)	SF (b)	SFM (r)	SFM (g)	SFM (b)
bird	4.498	4.467	6.400	8.37	8.76	9.93	0.0005	0.0005	0.0006
bulehills256	4.390	4.970	4.274	5.29	5.78	4.93	0.0005	0.0002	0.0001
columns	9.163	8.643	9.753	17.63	17.15	18.78	0.0014	0.0017	0.0015
couple	8.704	8.258	10.180	13.10	13.44	16.35	0.0013	0.0021	0.0030
8mile02_256	10.165	10.103	10.285	14.95	15.09	15.22	0.0008	0.0011	0.0014
marcie	13.648	13.804	14.880	18.95	19.19	20.43	0.0027	0.0041	0.0067
womanbaby256	4.728	5.880	6.747	8.94	10.44	11.56	0.0001	0.0002	0.0002
justinkelly256	20.549	20.499	20.664	26.11	25.99	25.63	0.0023	0.0034	0.0049
8mile01_256	23.067	22.574	21.777	14.95	15.09	15.22	0.0031	0.0044	0.0049
sail	28.801	26.407	26.729	31.30	28.71	28.33	0.0125	0.0090	0.0063
aerial02_256	36.128	36.793	35.569	41.90	42.88	41.45	0.0119	0.0123	0.0124
aerial01_256	23.067	22.574	21.777	28.72	28.57	27.50	0.0059	0.0063	0.0086
dallastwo256	22.995	22.622	19.297	33.39	32.54	28.98	0.0172	0.0173	0.0290
peppers	15.343	17.317	15.994	23.55	26.78	22.69	0.0022	0.0035	0.0096
balloons	4.568	5.788	6.608	8.77	11.55	12.22	0.0005	0.0008	0.0006
mountain256	37.339	36.537	38.445	41.84	42.16	46.30	0.0118	0.0130	0.0137
winter256	30.583	34.004	39.082	34.53	35.95	38.76	0.0611	0.0334	0.0143
lenacolor	15.431	20.437	18.708	20.15	26.29	22.15	0.0015	0.0077	0.0065
palmblades	43.620	40.905	5.855	45.90	42.01	17.10	0.0176	0.0112	0.0116
boy	17.224	19.327	16.909	21.07	23.19	20.50	0.0037	0.0043	0.0047

Synthetic Images:

Image	ImgGrad (r)	ImgGrad (g)	ImgGrad (b)	SF (r)	SF (g)	SF (b)	SFM (r)	SFM (g)	SFM (b)
snowwhite02	13.170	13.038	12.877	17.30	17.16	16.77	0.0018	0.0020	0.0023
toystory02_256	15.509	13.708	13.127	27.86	25.95	22.89	0.0028	0.0029	0.0039
toystory01_256	20.207	20.058	20.408	30.94	30.78	30.58	0.0146	0.0132	0.0140
mermaid01_256	27.914	25.652	25.112	48.96	46.94	43.87	0.0222	0.0199	0.0259
abugslife02_256	23.554	23.219	21.525	29.38	28.61	27.21	0.0177	0.0213	0.0496
bugsbunny02_256	19.067	17.293	18.069	34.97	33.80	32.69	0.0059	0.0057	0.0086
bugsbunny01_256	22.469	20.817	20.153	36.17	34.05	31.30	0.0077	0.0110	0.0121
geri02	19.283	18.917	18.478	26.62	26.51	25.72	0.0032	0.0056	0.0078
geri01	19.386	19.879	18.530	28.26	27.10	25.62	0.0016	0.0025	0.0027
mermaid02_256	21.742	22.670	22.776	44.11	47.91	47.00	0.0132	0.0114	0.0098
hercules01_256	22.649	19.603	19.411	42.35	38.14	37.85	0.0135	0.0097	0.0082
hercules02_256	25.677	23.960	22.361	45.90	44.36	39.83	0.0124	0.0118	0.0098
lionking01_256	14.287	12.725	12.501	22.24	21.12	20.92	0.0028	0.0028	0.0027
lionking02_256	8.397	7.601	7.428	12.83	12.90	13.20	0.0008	0.0007	0.0006
monsterinc01_256	16.645	16.653	16.862	23.88	23.90	23.52	0.0026	0.0026	0.0023
monsterinc02_256	19.879	19.417	19.425	26.26	25.51	24.61	0.0036	0.0036	0.0034
monsterinc03_256	20.202	20.694	21.394	24.36	24.82	25.16	0.0156	0.0070	0.0046
antz01	18.052	16.518	15.676	23.45	21.90	20.61	0.0059	0.0155	0.0263
antz02	12.742	10.985	10.604	20.90	19.22	18.46	0.0019	0.0034	0.0032
snowwhite01_256	9.884	9.653	9.563	13.59	13.64	13.04	0.0006	0.0006	0.0008

Clipart Images:

Image	ImgGrad (r)	ImgGrad (g)	ImgGrad (b)	SF (r)	SF (g)	SF (b)	SFM (r)	SFM (g)	SFM (b)
clipart09_256	10.607	17.276	18.794	52.01	66.37	69.23	0.0029	0.0057	0.0040
clipart06_256	6.713	7.013	7.148	38.88	39.77	40.29	0.0037	0.0038	0.0040
clipart10_256	12.919	11.854	11.622	51.54	47.97	47.23	0.0098	0.0093	0.0092
clipart05_256	6.540	8.457	9.054	37.48	37.09	39.89	0.0040	0.0048	0.0062
clipart07_256	23.709	28.926	34.360	64.72	74.55	88.52	0.0212	0.0259	0.0317
clipart08_256	8.407	17.619	8.919	32.67	39.74	33.58	0.0023	0.0034	0.0025
clipart01_256	3.015	2.799	4.926	10.80	11.42	19.43	0.0026	0.0070	0.0028
clipart03_256	7.483	3.755	8.852	25.61	16.14	32.05	0.0066	0.0058	0.0051
clipart02_256	7.729	3.770	2.963	34.26	22.92	21.29	0.0041	0.0094	0.0063
clipart04_256	39.101	35.075	14.474	97.29	87.02	52.67	0.0315	0.0285	0.0161
clipart11_256	18.964	20.183	19.881	60.36	62.65	61.90	0.0096	0.0104	0.0102
clipart12_256	12.436	20.023	15.245	56.31	71.46	62.35	0.0024	0.0079	0.0025
clipart13_256	0.000	1.681	1.681	0.00	20.70	20.70	0.0000	0.0000	0.0000
clipart14_256	8.903	8.210	7.717	47.65	45.76	42.55	0.0000	0.0000	0.0000
clipart15_256	7.658	10.724	10.724	44.19	52.29	52.29	0.0000	0.0000	0.0000
clipart16_256	7.658	6.241	3.860	44.19	39.89	31.37	0.0000	0.0044	0.0000
clipart17_256	21.320	21.320	13.451	72.22	72.22	58.57	0.0211	0.0211	0.0232
clipart18_256	14.416	11.812	12.074	51.81	44.17	42.86	0.0088	0.0052	0.0060
clipart19_256	18.058	15.781	10.164	61.39	56.51	48.19	0.0123	0.0117	0.0086
clipart20_256	7.658	9.229	9.229	44.19	48.51	48.51	0.0000	0.0032	0.0032

YCbCr Image Statistics

Natural Images:

Image	Y mean	Cb mean	Cr mean	Y Std. Dev	Cb Std. Dev	Cr Std. Dev	Y var.	Cb var.	Cr var.	Y skew	Cb skew	Cr skew	Y kurt	Cb kurt	Cr kurt
bird	114.66	7.22	-4.63	18.51	6.14	5.03	342.61	37.65	25.29	0.377	-3.473	5.305	15.619	13.624	49.911
bulehills256	119.55	50.81	-42.80	71.72	20.25	13.28	5143.84	410.21	176.35	0.366	-0.720	-0.042	-1.529	-0.867	-0.952
columns	91.68	1.65	3.59	63.65	6.37	6.29	4051.52	40.61	39.61	0.667	0.214	2.776	-0.569	0.508	9.963
couple	83.33	-5.60	12.33	50.96	16.14	18.62	2597.02	260.52	346.60	1.021	-0.126	0.586	0.780	1.771	0.061
8mile02_256	105.66	-9.96	10.74	61.92	7.41	8.16	3833.60	54.96	66.63	0.398	-0.702	0.075	-0.604	0.227	-0.727
marcie	100.78	-12.73	12.59	45.61	6.97	7.62	2080.57	48.54	58.11	0.379	-0.641	-0.357	0.514	1.475	-0.571
womanbaby256	219.01	-12.46	17.03	37.79	11.82	14.93	1427.86	139.63	222.92	-1.131	-0.434	0.330	0.856	-1.365	-1.335
justinkelly256	115.32	-16.46	20.25	80.40	8.99	10.54	6463.91	80.90	111.00	0.150	-0.065	0.271	-1.504	-0.378	-0.592
8mile01_256	94.22	-3.83	10.85	54.49	11.98	11.24	2969.00	143.59	126.36	0.309	-0.263	0.471	-1.132	-0.910	-0.539
sail	110.62	12.77	-9.05	36.51	12.44	11.42	1332.63	154.64	130.47	0.937	-1.408	1.839	0.219	5.881	4.781
aerial02_256	117.38	-2.05	-1.77	60.89	6.37	10.00	3707.75	40.64	99.95	0.064	-0.703	2.623	-1.126	4.123	24.208
aerial01_256	118.31	-9.33	1.76	49.08	7.79	13.05	2408.96	60.69	170.36	-0.176	-0.244	1.197	-0.399	0.351	6.703
dallastwo256	74.57	-11.32	4.62	54.01	15.55	7.95	2917.32	241.74	63.26	1.073	-1.498	1.475	0.327	2.994	7.071
peppers	118.70	-29.64	19.62	53.81	12.95	33.66	2895.61	167.66	1132.76	-0.238	-0.124	0.488	-0.878	-0.229	-1.223
balloons	119.13	10.97	-8.21	38.72	29.13	29.96	1499.43	848.73	897.68	-0.313	-1.594	1.244	-1.165	2.618	0.283
mountain256	137.11	2.51	3.21	67.79	17.35	10.77	4595.13	300.94	115.96	0.272	1.906	-0.980	-1.112	3.561	0.545
winter256	63.84	35.49	-22.18	43.29	12.11	7.85	1873.67	146.73	61.63	1.695	0.018	-0.251	2.533	-0.482	-0.421
lenacolor	123.97	-10.52	40.09	47.87	13.65	12.85	2291.74	186.42	165.12	-0.081	0.436	-0.205	-0.847	-0.530	-0.851
palmblades	95.66	-47.96	2.55	44.10	23.70	6.80	1945.17	561.86	46.24	0.649	-0.457	2.541	0.329	-0.303	8.960
boy	104.85	-8.12	-1.65	63.60	20.15	15.89	4045.29	405.91	252.61	0.840	-1.659	-0.013	-0.509	6.318	-0.591

Synthetic Images:

Image	Y mean	Cb mean	Cr mean	Y Std. Dev	Cb Std. Dev	Cr Std. Dev	Y var.	Cb var.	Cr var.	Y skew	Cb skew	Cr skew	Y kurt	Cb kurt	Cr kurt
snowwhite02	79.93	-3.03	2.65	40.72	5.37	4.41	1657.76	28.79	19.41	0.904	-0.834	1.070	0.613	1.206	4.415
toystory02_256	151.64	-19.47	9.79	80.36	19.28	16.92	6457.11	371.80	286.21	-0.157	-0.683	1.399	-1.697	0.380	1.521
toystory01_256	82.57	-1.16	-2.64	57.06	9.85	8.74	3255.82	97.05	76.31	0.735	-0.265	0.255	-0.031	0.614	1.143
mermaid01_256	117.17	-9.74	-6.44	56.11	26.38	40.12	3148.45	695.74	1609.40	0.536	-0.828	0.365	-0.874	2.186	0.196
abugslife02_256	51.59	-13.37	0.97	39.61	14.61	13.14	1568.82	213.44	172.56	1.153	-1.226	1.877	1.134	3.017	4.427
bugsbunny02_256	159.33	-23.83	9.45	58.28	32.80	32.91	3396.10	1076.14	1082.98	-0.914	0.126	0.176	-0.537	-1.009	-0.853
bugsbunny01_256	111.68	-8.78	14.57	69.50	18.90	20.06	4829.83	357.12	402.59	0.250	-0.337	1.214	-1.223	-1.191	2.106
geri02	112.11	-15.28	20.62	62.80	9.08	18.61	3943.46	82.43	346.38	0.319	0.097	1.214	1.214	-0.569	1.138
geri01	120.22	-16.47	26.30	73.54	18.21	32.07	5408.26	331.50	1028.40	0.053	0.114	0.437	-1.216	-1.098	-0.679
mermaid02_256	159.36	3.14	-13.36	64.17	25.16	38.64	4117.57	633.18	1492.94	-1.117	-0.204	0.944	-0.157	-1.425	2.387
hercules01_256	134.51	-0.64	-10.60	63.35	34.77	27.99	4013.74	1208.88	783.38	0.126	-0.256	1.288	-0.955	-0.727	1.165
hercules02_256	148.52	-5.16	-1.84	59.04	31.43	26.90	3485.91	987.88	723.38	-0.322	0.619	-0.272	-0.520	0.110	0.031
lionking01_256	98.90	-8.24	6.49	53.88	21.42	29.30	2902.61	458.73	858.62	0.216	0.193	-0.094	-1.512	-0.912	-1.504
lionking02_256	99.86	-1.51	-0.93	34.44	25.27	25.16	1186.14	638.50	633.18	-0.240	0.077	0.258	-0.472	-1.438	-0.985
monsterinc01_256	166.61	6.91	-2.35	55.45	15.04	12.00	3074.67	226.18	144.00	-0.038	-0.501	-0.902	-1.016	0.245	0.756
monsterinc02_256	156.02	1.94	0.29	56.86	20.26	10.15	3233.50	410.44	103.02	0.235	-0.419	-0.535	-0.870	-0.670	0.671
monsterinc03_256	92.11	21.34	-21.02	49.04	8.07	11.20	2404.44	65.09	125.53	0.681	-1.088	-0.084	-0.586	1.643	-1.182
antz01	68.43	-15.42	23.39	31.02	6.70	16.86	962.46	44.84	284.17	0.765	-0.761	0.795	1.236	0.975	-0.128
antz02	119.04	-10.43	24.87	56.73	18.05	13.26	3217.96	325.96	175.93	-0.078	-0.835	1.191	-0.765	1.249	1.567
snowwhite01	91.25	-3.82	2.67	48.53	9.83	9.95	2355.03	96.70	99.10	0.901	-0.151	2.425	-0.002	1.163	6.165

Clipart Images:

Image	Y mean	Cb meen	Cr mean	Y Std. Dev	Cb Std. Dev	Cr Std. Dev	Y var.	Cb var.	Cr var.	Y skew	Cb skew	Cr skew	Y kurt	Cb kurt	Cr kurt
clipart09_256	135.87	10.76	37.70	99.92	69.26	63.36	9984.94	4796.65	4014.49	0.240	0.668	0.665	-1.770	-0.469	-1.461
clipart06_256	197.22	0.14	-0.24	104.20	3.18	5.37	10858.04	10.11	28.81	-1.319	27.852	-23.073	-0.193	868.022	539.302
clipart10_256	184.30	-2.89	5.38	78.75	3.18	5.93	6201.30	10.08	35.11	-1.121	-0.194	0.194	0.625	-1.963	-1.963
clipart05_256	209.93	-17.63	17.49	52.21	21.90	22.47	2726.00	479.81	504.82	-1.551	-0.993	1.204	3.989	-0.183	0.572
clipart07_256	156.20	9.21	-7.14	91.07	21.52	17.14	8293.82	463.11	293.80	-0.410	1.886	-2.123	-1.055	3.978	4.458
clipart08_256	146.42	-6.85	-10.02	113.06	13.70	19.48	12781.97	187.71	379.59	-0.157	-1.277	-2.416	-1.868	3.556	9.998
clipart01_256	38.35	16.45	9.08	26.32	24.34	28.10	692.81	592.57	789.79	1.429	0.181	-1.094	3.243	-1.292	2.136
clipart03_256	52.01	30.81	3.31	56.50	31.00	15.71	3191.79	961.19	246.95	1.195	0.249	1.333	0.118	-1.344	1.766
clipart02_256	57.92	-21.09	50.36	44.01	19.47	38.93	1936.71	379.22	1515.76	0.452	-0.478	-0.107	-0.028	-1.125	-1.581
clipart04_256	182.13	-35.21	11.41	89.50	50.07	16.65	8009.95	2507.28	277.29	-1.341	-0.748	0.748	0.265	-1.439	-1.439
clipart11_256	187.78	-0.01	-0.66	103.67	2.87	8.29	10746.51	8.24	68.67	-1.096	13.776	-14.693	-0.607	199.475	217.010
clipart12_256	136.92	20.52	31.64	102.52	65.77	61.88	10511.01	4325.71	3828.82	0.209	0.839	0.862	-1.811	-0.930	-1.164
clipart13_256	230.20	-5.97	17.69	61.79	14.87	44.07	3818.04	221.23	1942.43	-2.090	-2.090	2.090	2.369	2.369	2.369
clipart14_256	109.02	1.93	39.94	102.26	37.92	62.11	10456.85	1438.18	3857.16	0.584	0.122	0.694	-1.391	-1.471	-1.505
clipert15_256	100.25	58.95	10.55	99.14	72.58	53.19	9828.56	5267.77	2828.83	0.855	-0.195	1.669	-1.144	-1.799	0.969
clipart16_256	96.16	-45.41	107.83	58.50	26.42	44.22	3422.00	697.81	1955.28	1.883	-1.780	-1.827	2.390	4.778	1.399
clipart17_256	134.31	-34.06	5.54	114.08	60.86	9.90	13014.18	3703.48	97.94	-0.303	-0.838	-0.838	-1.857	-1.188	-1.188
clipart18_256	212.10	-6.78	-13.18	60.82	11.77	17.77	3699.55	138.47	315.60	-1.670	-1.429	-1.492	2.284	5.205	2.375
clipart19_256	163.94	-7.98	10.40	111.37	19.79	22.27	12403.52	391.58	496.09	-0.545	-3.880	2.647	-1.571	17.465	8.131
clipart20_256	171.11	-18.35	54.37	92.42	21.28	63.06	8542.21	452.87	3976.23	-0.245	-0.297	0.297	-1.805	-1.912	-1.912

YCbCr Entropy Measures

Natural Images:

	Y			Cb			Cr		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
bird	7.8455	4.8210	6.5996	4.0000	2.7994	4.3005	6.2479	0.8428	0.8322
bulehills256	7.7879	7.1411	6.9843	6.2095	5.1897	5.2306	0.0000	0.0000	0.0000
columns	7.8329	7.4748	7.6922	4.9542	2.9685	3.6772	5.5236	3.4113	4.9402
couple	7.9248	7.2844	7.7105	5.3576	1.9697	2.0215	6.1699	4.8401	5.7324
8mile02_256	7.9484	7.6807	7.5784	4.3219	0.1703	0.1906	5.6724	4.5733	5.0661
marcie	8.0000	7.4006	7.7706	3.8074	0.0849	0.0885	5.6147	4.5883	6.3331
womanbaby256	7.5699	6.1789	5.9075	1.5850	0.0424	0.0715	5.8826	4.9608	4.8553
justinkelly256	7.9887	7.7659	7.7340	3.1699	0.0838	0.1089	5.8826	5.3638	6.2712
8mile01_256	7.8329	7.5273	7.6147	4.8074	2.8478	2.7634	5.9069	4.4193	4.8450
sail	7.5850	6.9794	7.6341	5.7004	4.6782	5.8408	5.9773	1.3882	1.3812
aerial02_256	7.9658	7.7308	7.8624	4.3219	2.0332	2.6060	6.5078	2.3883	3.0300
aerial01_256	7.9830	7.4123	7.7274	4.5850	0.7705	0.8799	6.6724	3.3758	3.9099
dallastwo256	7.8704	7.4153	7.7642	4.0000	1.5073	1.6735	6.3038	3.9592	5.5910
peppers	7.9009	7.5731	7.7620	2.8074	0.0114	0.0140	6.4094	4.5528	4.6316
balloons	7.3038	6.7985	7.3245	5.5850	4.2004	4.5763	6.1699	2.2457	2.1388
mountain256	7.9887	7.5707	7.5269	6.1699	2.8225	3.0739	5.0875	3.8443	5.0423
winter256	7.9542	6.9416	7.5081	6.1085	5.6104	6.3366	1.5850	0.0014	0.0011
lenacolor	7.7211	7.4425	7.8193	5.0000	1.8068	1.9048	6.2668	5.8087	7.1356
palmblades	7.9425	7.4183	7.7852	4.7004	0.0464	0.0345	5.7004	2.8055	4.0484
boy	7.9069	7.4764	7.6901	4.5236	1.7394	1.9472	5.3923	3.2838	3.7095

Synthetic Images:

	Y			Cb			Cr		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
snowwhite02	7.9366	7.1954	7.5736	3.4594	1.2520	1.5084	5.1293	3.0677	4.0263
toystory02_256	7.8580	6.6446	6.7628	4.8074	1.1272	1.0107	6.2668	4.2497	4.3022
toystory01_256	8.0000	7.5481	7.4985	5.3576	2.5959	2.6765	5.5850	2.1055	2.3149
mermaid01_256	7.9830	7.4224	7.4233	6.0224	2.5413	2.5319	7.0000	3.8749	3.6571
abugslife02_256	7.8887	6.9869	7.4359	5.2479	0.7236	0.7115	6.0444	2.7084	2.6965
bugsbunny02_256	8.0000	5.7005	5.6869	5.7004	1.5979	1.3465	6.7944	3.3038	3.1779
bugsbunny01_256	7.9944	7.6401	7.5889	5.0444	2.7210	2.6001	6.6439	4.8456	4.9390
geri02	7.9944	7.8022	7.8046	3.3219	0.1256	0.1905	6.5236	5.5457	6.1291
geri01	8.0000	7.2339	7.0578	4.4594	1.1484	1.0972	6.9189	5.7067	6.0645
mermaid02_256	7.9484	6.9094	6.9267	5.7279	3.5948	3.0542	6.7549	2.0971	1.7808
hercules01_256	8.0000	7.2039	6.9727	5.7814	3.5199	3.0536	6.5546	2.1082	1.8786
hercules02_256	8.0000	7.6262	7.5994	6.3399	3.4179	3.1183	6.4429	3.3947	3.3090
lionking01_256	7.7211	7.0216	7.2522	5.0000	2.2923	1.9345	5.8074	3.9316	3.8158
lionking02_256	7.5850	6.7831	7.1861	5.4263	3.4889	3.1747	5.8826	3.4234	3.1782
monsterinc01_256	7.9484	7.5335	7.5815	5.2854	4.1095	4.3144	4.9069	3.0380	3.5560
monsterinc02_256	7.9542	7.4775	7.5790	5.3219	3.6153	3.6701	4.9069	3.2505	3.8579
monsterinc03_256	7.9600	7.3962	7.5790	5.2479	4.8050	5.7265	3.3219	0.0598	0.0592
antz01	7.8265	6.9121	7.3884	3.0000	0.0356	0.0308	6.3750	5.7831	6.1766
antz02	7.9484	7.6804	7.6182	4.9069	2.1885	2.0730	6.3923	5.4068	6.2293
snowwhite01	7.7004	7.2697	7.5038	5.0000	1.6155	1.9056	5.6439	2.5715	3.1328

Clipart Images:

	Y			Cb			Cr		
Image	H0	H1	H2	H0	H1	H2	H0	H1	H2
clipart09_256	2.3219	1.8667	1.4171	1.0000	0.7778	0.5434	1.5850	1.1106	0.8471
clipart06_256	2.3219	0.9697	0.5570	1.5850	0.0252	0.0185	0.0000	0.0000	0.0000
clipart10_256	1.5850	1.3922	1.0416	0.0000	0.0000	0.0000	1.0000	0.9933	0.8201
clipart05_256	4.5236	3.2774	2.1141	0.0000	0.0000	0.0000	4.3923	3.0916	2.0109
clipart07_256	7.0334	3.1599	2.1290	5.5236	1.8292	1.3600	5.3923	0.1358	0.1176
clipart08_256	7.9658	3.3977	2.8319	5.4263	0.1337	0.1110	5.5236	0.0372	0.0269
clipart01_256	7.0334	4.8280	3.8926	6.1085	3.5382	2.5710	5.7814	4.0415	2.6866
clipart03_256	7.3576	4.8583	3.3884	6.5392	4.3665	3.0762	5.1699	2.5788	1.7467
clipart02_256	7.1699	4.9634	3.2648	0.0000	0.0000	0.0833	6.8202	4.7385	3.1397
clipart04_256	2.0000	1.9177	1.4799	0.0000	0.0000	0.0000	1.0000	0.9094	0.8116
clipart11_256	6.8704	1.6388	1.1329	4.9542	0.0628	0.0491	0.0000	0.0000	0.0285
clipart12_256	2.0000	1.7006	1.3204	1.0000	0.8251	0.6173	1.0000	0.8690	0.6628
clipart13_256	1.0000	0.5809	0.3801	0.0000	0.0000	0.0000	1.0000	0.5809	0.3801
clipart14_256	2.0000	1.7314	1.3124	1.0000	0.9074	0.6509	1.0000	0.9190	0.6824
clipart15_256	2.0000	1.5902	1.2783	1.0000	0.9990	0.8265	1.0000	0.8511	0.4555
clipart16_256	2.0000	0.8983	0.9824	0.0000	0.0000	0.0000	1.5850	0.8147	0.9237
clipart17_256	2.0000	1.8522	1.3225	1.0000	0.5671	0.4031	1.0000	0.8738	0.6644
clipart18_256	4.0875	2.6753	1.8349	1.0000	0.4650	0.4843	1.0000	0.3432	0.2503
clipart19_256	3.0000	1.9703	1.3221	0.0000	0.0000	0.0000	2.8074	1.3418	0.9174
clipart20_256	1.5850	1.1543	0.8938	0.0000	0.0000	0.0000	1.0000	0.9843	0.7649

Image Gradient, Spatial Frequency, and Spectral Flatness Measure Results (YCbCr)

Natural Images:

Image	ImgGrad (Y)	ImgGrad (Cb)	ImgGrad (Cr)	Sf (Y)	SF (Cb)	SF (Cr)	SFM (Y)	SFM (Cb)	SFM (Cr)
bird	4.077	2.634	1.686	8.370	2.815	2.430	0.0004	0.0137	0.0167
bulehills256	4.574	0.814	1.126	5.321	1.339	1.542	0.0002	0.0001	0.0001
columns	8.539	2.247	2.164	17.152	2.720	3.321	0.0015	0.0211	0.0271
couple	8.113	2.962	2.459	13.018	4.249	3.821	0.0018	0.0076	0.0031
8mile02_256	10.028	1.187	1.348	14.970	1.542	1.779	0.0010	0.0011	0.0012
marcie	13.552	3.003	2.459	19.033	3.187	2.665	0.0037	0.0073	0.0046
womanbaby256	5.519	1.176	1.397	9.915	1.851	2.284	0.0002	0.0006	0.0005
justinkelly256	20.301	2.323	2.734	25.801	2.637	3.079	0.0031	0.0011	0.0009
8mile01_256	16.389	1.796	1.834	21.027	2.214	2.608	0.0040	0.0023	0.0018
sail	26.544	5.539	5.010	28.756	6.602	5.956	0.0093	0.0143	0.0176
aerial02_256	35.865	4.066	5.085	41.894	4.556	6.653	0.0119	0.0633	0.0506
aerial01_256	22.256	2.188	3.348	27.973	3.108	5.630	0.0063	0.0057	0.0150
dallastwo256	21.739	6.922	4.599	31.407	9.423	6.252	0.0171	0.0402	0.0767
peppers	15.739	4.232	5.422	24.209	5.343	7.468	0.0034	0.0017	0.0011
balloons	4.533	3.693	2.919	8.745	6.607	5.714	0.0005	0.0041	0.0030
mountain256	36.736	6.275	4.489	42.169	6.626	4.741	0.0126	0.0241	0.0291
winter256	33.097	6.834	4.637	35.410	6.732	4.792	0.0348	0.0035	0.0059
lenacolor	17.325	6.786	6.662	22.882	6.974	6.998	0.0040	0.0267	0.0046
palmblades	37.060	20.910	6.733	38.278	21.804	8.486	0.0125	0.0155	0.1732
boy	17.834	3.864	4.162	21.811	4.355	4.092	0.0040	0.0050	0.0104

Synthetic Images:

Imaga	ImgGrad (Y)	ImgGrad (Cb)	ImgGrad (Cr)	Sf (Y)	SF (Cb)	SF (Cr)	SFM (Y)	SFM (Cb)	SFM (Cr)
snowwhite02	12.951	0.952	0.813	17.112	1.262	1.169	0.0019	0.0027	0.0038
toystory02_256	13.564	3.408	3.401	25.463	5.806	5.777	0.0029	0.0018	0.0040
toystory01_256	19.781	2.949	2.732	30.534	4.162	3.866	0.0136	0.0117	0.0120
mermaid01_256	24.530	5.551	7.850	45.315	9.370	13.269	0.0214	0.0115	0.0107
abugslife02_256	22.870	3.799	2.549	28.458	4.309	2.982	0.0224	0.0023	0.0023
bugsbunny02_256	16.643	4.344	4.691	32.975	7.153	7.979	0.0060	0.0018	0.0018
bugsbunny01_256	20.636	3.611	3.743	33.775	5.627	6.450	0.0099	0.0040	0.0038
geri02	18.660	1.851	3.270	26.160	2.159	4.124	0.0048	0.0013	0.0020
geri01	18.450	4.328	7.327	25.746	5.477	9.482	0.0021	0.0023	0.0023
mermaid02_256	21.209	3.851	6.287	45.140	6.134	12.184	0.0116	0.0041	0.0090
hercules01_256	19.586	5.073	4.617	38.305	8.378	8.362	0.0107	0.0045	0.0050
hercules02_256	23.458	5.717	4.324	43.631	9.052	6.717	0.0119	0.0061	0.0039
lionking01_256	12.750	2.266	2.862	21.107	3.026	3.877	0.0029	0.0009	0.0006
lionking02_256	7.490	1.961	1.993	12.612	2.576	2.792	0.0007	0.0004	0.0004
monsterinc01_256	16.368	2.663	2.364	23.584	3.719	3.080	0.0025	0.0026	0.0035
monsterinc02_256	19.157	3.463	3.054	25.272	4.662	3.530	0.0035	0.0034	0.0097
monsterinc03_256	20.408	2.227	2.473	24.503	2.945	3.155	0.0080	0.0010	0.0012
antz01	16.634	1.915	2.599	21.992	2.444	3.357	0.0116	0.0025	0.0007
antz02	11.002	2.807	2.775	19.209	4.190	3.883	0.0028	0.0025	0.0010
snowwhite01_256	9.486	1.005	1.089	13.440	1.529	1.738	0.0006	0.0008	0.0012

Clipart Images:

Imaga	ImgGrad (Y)	ImgGrad (Cb)	ImgGrad (Cr)	Sf (Y)	SF (Cb)	SF (Cr)	SFM (Y)	SFM (Cb)	SFM (Cr)
clipart09_256	14.606	5.748	5.896	54.852	31.495	26.614	0.0033	0.0256	0.0013
clipart06_256	6.938	0.118	0.161	39.335	3.160	4.258	0.0038	0.1082	0.0668
clipart10_256	12.132	0.297	0.554	48.880	1.377	2.569	0.0095	0.0154	0.0154
clipart05_256	7.882	1.851	2.454	35.760	9.186	11.756	0.0043	0.0153	0.0243
clipart07_256	27.888	5.170	4.193	71.947	16.384	13.363	0.0248	0.0708	0.0757
clipart08_256	13.696	3.313	4.479	35.012	7.720	10.952	0.0027	0.0182	0.0192
clipart01_256	2.467	1.763	1.866	10.100	6.831	6.525	0.0036	0.0040	0.0062
clipart03_256	5.160	3.145	2.348	18.954	10.388	8.621	0.0054	0.0060	0.0151
clipart02_256	4.764	1.288	2.704	24.282	5.526	11.804	0.0051	0.0049	0.0042
clipart04_256	33.733	12.143	4.005	83.732	35.990	11.883	0.0275	0.0559	0.0560
clipart11_256	19.689	0.355	0.697	61.269	2.968	8.618	0.0100	0.1461	0.1476
clipart12_256	17.218	5.414	5.069	60.962	23.748	24.047	0.0048	0.0059	0.0057
clipart13_256	1.180	0.284	0.841	14.533	3.493	10.352	0.0000	0.0000	0.0000
clipart14_256	8.363	1.275	2.344	43.175	7.660	16.998	0.0000	0.0000	0.0000
clipart15_256	9.810	3.520	2.367	46.378	19.867	15.868	0.0000	0.0000	0.0000
clipart16_256	6.383	1.430	2.896	35.744	9.706	18.669	0.0013	0.0015	0.0005
clipart17_256	20.373	5.015	0.816	68.954	24.172	3.931	0.0214	0.0186	0.0186
clipart18_256	12.501	1.623	1.968	45.607	6.616	8.413	0.0061	0.0350	0.0217
clipart19_256	15.770	3.219	3.571	54.835	13.237	14.472	0.0109	0.0549	0.0484
clipart20_256	8.761	0.901	2.669	44.150	6.226	18.448	0.0016	0.0039	0.0039

APPENDIX 3

Publications:

1. Yap,V.V., Comley, R.A., "A Segmentation-based Wavelet Compression Scheme for Still Images", *IASTED International Conference on Signal and Image Processing (SIP2004)*, Honolulu, 2004

2. Yap,V.V., Rahman, S, "Performance of Wavelet-based Image Compression on Synthetic Images", *International Conference on Robotics, Vision, Information And Signal Processing*, IEEE Malaysia, Penang, Malaysia , January 2000

3. Yap,V.V., Rahman, S, "The Implications of Image Statistics and Image Features on Coding Performance of Synthetic Images", *SCI 2003*, Orlando, Florida, July 2003

A Segmentation-based Wavelet Compression Scheme for Still Image

Vooi Voon Yap[1] , Richard Comley[1]
[1]School of Computing Science
Middlesex University,
White Hart Lane
London N17 8HR, United Kingdom,
Tel: +44-02084114235,
E-mail:v.yap@mdx.ac.uk
E-mail:r.comley@mdx.ac.uk

ABSTRACT
There has been much recent interest in the use of wavelet-based image compression schemes. They offer various advantages, not least the elimination of the need to segment an image prior to using the Discrete Cosine Transform (DCT). It is shown in this paper, however, that image segmentation and separation can produce an equivalent performance to more conventional schemes and can offer certain advantages. A wavelet-based image compression scheme is presented in which an image is first partitioned into high and low frequency segments and different wavelets are then applied using the 2-D discrete wavelet transformation (DWT).

KEY WORDS
wavelet-based, edge-detection, segmentation.

I. Introduction

The advent of picture messaging on mobile telephones, Personal Digital Assistant (PDA) technology and other wireless based image services has shifted the balance of priority for image compression schemes. Historically, the output medium for decompressed images was a high resolution computer monitor that demanded a good quality image. Image quality can be broadly interpreted to mean a good Peak Signal to Noise Ratio (PSNR). The capabilities of the display screen on the majority of mobile devices combined with the cost of transmission bandwidth has shifted the emphasis away from the PSNR towards the level of compression, frequently expressed as a direct compression ratio or via a bits per pixel (bpp) value.

For many years the Joint Photographic Experts Group (JPEG), a DCT-based compression scheme, has been the still image compression standard of choice. However, in recent years wavelet-based compression has become popular because such compression schemes allow high compression ratios while maintaining good image quality. The popularity and success of wavelets has prompted its inclusion in the JPEG2000 standard. A variation on conventional wavelet-based image compression schemes
is presented here. The proposed technique exploits the variable frequency characteristics of an image by applying different wavelet filters to the low and high frequency elements. It is shown that this can offer an improvement in the compression ratio for certain types of image, with little overall loss in PSNR.

2. Image Characteristics

A typical grey-scale image is comprised of pixels that are correlated and therefore contain redundant information. Significant compression can be achieved by exploiting these redundancies. A detailed examination of the frequency characteristics of the image pixels can provide useful information pertaining to the potential compression of the image concerned. The frequency characteristics can be better viewed by splitting the image into blocks and computing the Fourier log power spectrum of each block [3].

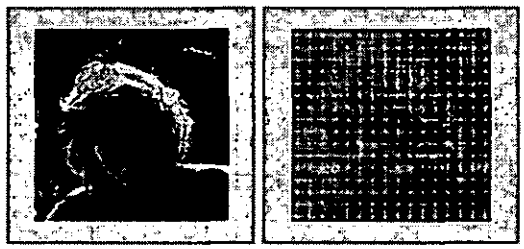


Figure 1 Original image (source: Kodak) and Fourier log pawer spectrum image

The above images indicate a wide variation in spectra. The bright center in the blocks, indicate a low frequency component. The blocks that contain the hair produce a spread of energy at all frequencies. Spectra with lines normal to the edges indicate the presence of dominant edges

(high frequencies), for example, the transition between the shoulder and the background.

3. Conventional Wavelet Compression System

In recent years researchers have focused on the utilization of discrete wavelet transforms (DWT) in digital image compression. The advantage of the DWT is that, in contrast to the discrete cosine transform (DCT), it does not require the image to be divided into blocks, but analyses the image as a whole.

The following is a conventional lossy wavelet compression system. The image is first decomposed into wavelet coefficients by a forward discrete wavelet transform. The result is four different sets of coefficients or subband images, that is, the approximation coefficients and the three sets of detail coefficients. Many of the resultant wavelet coefficients are close to zero. These wavelet coefficients are then quantised. Finally, the wavelet coefficients are encoded and an output bit stream is produced.

Figure 2 Typical wavelet compression scheme

4. The Discrete Wavelet Transform

The key to any wavelet-based compression scheme is the wavelet transform. This section provides a short review of the wavelet transform algorithm. The 1D wavelet transform decomposition and reconstruction algorithm basically utilizes a pair of complementary filter banks. Each filter consists of a low-pass filter ($H(z)$) and a high-pass filter ($G(z)$). Figure 3 describes a single stage, two-channel filter bank.

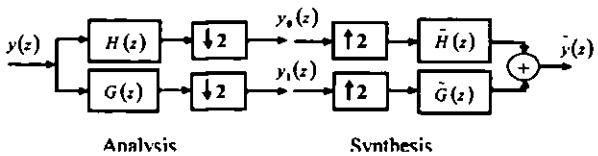


Figure 3 Single stage two-channel filter bank

The input $y(z)$ is filtered by a low-pass filter $H(z)$ and down-sampled by a factor of two to produce $y_0(z)$. Output $y_1(z)$ is obtained by filtering $y(z)$ with a high-pass filter $G(z)$ and down-sampled by a factor of two. The 1D wavelet transform is well-documented and can be found in [3], [4], [5] and hence will not be discussed further. Instead, this paper will focus on how the 2D wavelet transform is used to decompose and reconstruct an image.

In two dimensions, the decomposition is achieved by applying the 1D transform in the horizontal and vertical directions as shown in Figure 4.

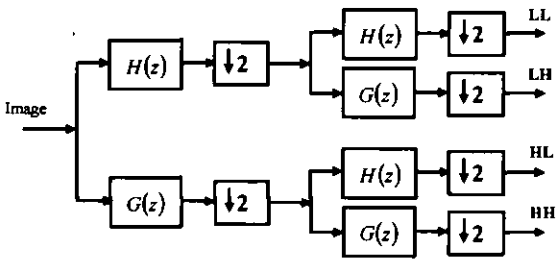


Figure 4 Level one of the wavelet transform

The outputs LL, LH, HL, and HH are sub-band images. LL is the lowest sub-band image and it represents a smaller low-resolution version of the original image. The other sub-band images LH, HL, and HH are high-pass samples and they represent a smaller residual version of the original image.

At the reconstruction stage the sub-band images are recombined so that the original image can be reconstructed. The reconstruction process is depicted in Figure 5.

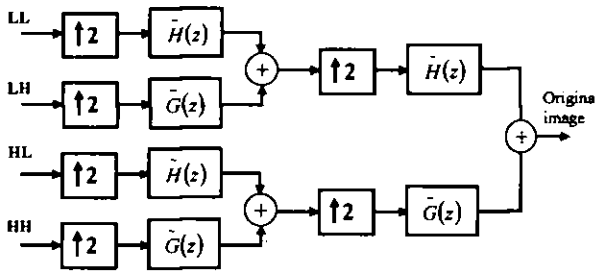


Figure 5 Wavelet transform reconstruction process

Reconstruction is achieved by up-sampling the lower resolution images and passing them through the synthesis filters, $\tilde{H}(z)$ and $\tilde{G}(z)$.

5. Method

The above approach uses the same wavelet filter to compress the whole image. This paper

proposes a compression scheme based on the segmentation of an image into its low and high frequency parts. A different wavelet filter is then applied to these different elements of the image.

An edge-detector algorithm was used to segment the grey-scale image into low frequency and high frequency regions. Since edges are usually sharp changes rather than slow-varying changes, an edge-detector can be used to identify fast gradient changes. The gradient change of
an image can be derived by computing its first derivative by finding the numerical approximation of the difference in each pixel. The gradient change can be computed by:

$$\sqrt{[I(r,c)-I(r-1,c-1)]^2+[I(r,c-1)-I(r-1,c)]^2}$$

$I(r,c)$ denotes an image pixel at row (r) and column (c).
Typically, an approximate magnitude is computed using:

$$|I(r,c)-I(r-1,c-1)|+|I(r,c-1)-I(r-1,c)|$$

This form of the equation is generally preferred because it is much faster to compute [2].

Edge-detectors work on the basis that edge information in an image can be located by examining the relationship of a particular pixel and the surrounding pixels. If there is a wide variation of grey levels surrounding a pixel then an edge is present. On the other hand, if the grey levels are similar then there is no edge present at that point.

The following is a modified version of an edge detection algorithm scheme first presented by Liu (2004) [1]. A Roberts operator, chosen because it works best with grey-scale images [2], is used to extract the edges. The Roberts operator mask is defined as:

$$\text{row mask} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{column mask} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The Roberts operator convolution masks are convolved with the grey-scale image to produce an edge-detected image, which is basically a binary image. The resultant edge-detected image is then divided into blocks; values of 8 x 8, 16 x 16 and 32 x 32 were used in this study. The number of black pixels in each block is counted. If the block contains more than 10 black pixels, then it is considered to be a high frequency region. However, if the block contains less than 10 black pixels, then that region is said to be a low frequency region. Figure 6 provides an example of its operation.

Figure 6 Segmented images

The low frequency and high frequency images were then compressed separately using a different wavelet filter for each image. The following is a comparison of the coding

performance using a single wavelet compression scheme and the dual wavelet method presented above.

6. Results

The segmentation scheme described above was implemented using Matlab, using two of the wavelets available in that package. The first set of results (Figure 7) gives a comparison of the output from the encoder for a single wavelet and the dual wavelet scheme. The number of bytes coming from the encoder has been used to provide an absolute measure of performance rather than the compression ratio, as is more normal. The reason for this is that the dual wavelet method produces two image outputs that must be added to produce the total data size. The bior4.4 wavelet was used for the single wavelet method at a decomposition level of 3. For the dual wavelet method, the bior4.4 was again used at level 3 for the low frequency image and a haar wavelet at a decomposition level of 4, for the high frequency image.

Image	Encoder output (single)	Encoder output (dual)
gmandrill	19125	14704
gmarcie	6849	5850
gpalmblades	15381	13686
gbarbara	11504	9286
gboat	8162	7399
ggoldhill	6889	5970

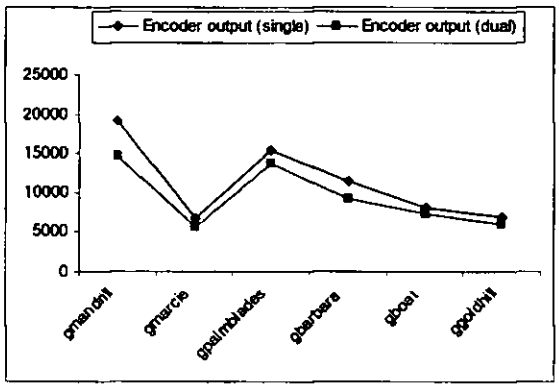


Figure 7 Comparison of resultant compressed file sizes for the single and dual wavelet methods.

From figure 7 it is apparent that there is some improvement in compression as the total number of bytes coming from the encoder is smaller for the dual wavelet method. However, a direct comparison cannot be made as the effect of the application of the dual wavelet is to reduce the overall PSNR by approx. 2dB for each of the images.

Image	PSNR (single)	PSNR (dual)
gmandrill	29.20	27.14
gmarcie	32.93	30.84
gpalmblades	29.31	27.77
gbarbara	31.29	28.55
gboat	31.78	30.17
ggoldhill	31.52	29.53

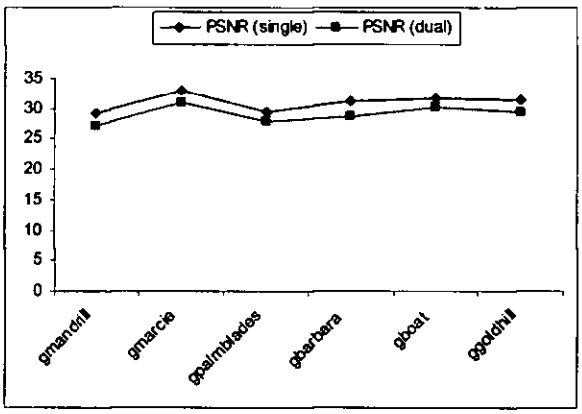


Figure 8 Reduction in PSNR caused by the dual wavelet method.

In order to provide a direct comparison, the PSNR for the single wavelet method was set to the same level as that achieved by the dual method, and the number of bytes produced by the encoder was recomputed. The results are shown in Figure 9.

Image	Encoder output (single)	Encoder output (dual)
gmandrill	15574	14704
gmarcie	5254	5850
gpalmblades	12954	13686
gbarbara	8753	9286
gboat	6580	7399
ggoldhill	4972	5970

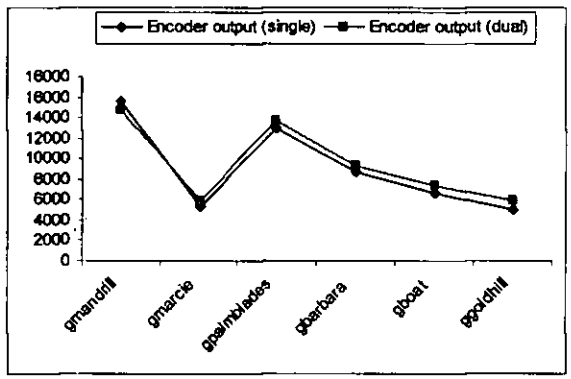


Figure 9 Comparison of the single and dual wavelet methods for fixed PSNR levels.

7. Conclusions and Future Work

Initial results indicate that some improvement in compression can be achieved through the use of a combination of wavelets with little loss in the quality of the reconstructed image. This could prove to be a very significant factor for mobile devices, where bandwidth costs can be high and small reductions in PSNR would have little, if any, impact.

Current work is concentrating on extending the technique to colour images where it is anticipated that more significant improvements may be possible. Work is also underway in order to classify regions of an image based on their frequency content, and then to match a small family of wavelets to these regions in order to optimise overall performance. The ultimate aim is to develop a simple expert system to perform the initial analysis on an image and then to assign the most appropriate combination of wavelets for its compression.

8. References

[1] Liu, N, "Mini-Project #1: Spatial Masking", Online.Google. http://people.cornell.edu/pages/nwl2/project1-webpage/project1-webpage.htm. View date: March, 2004.

[2] Umbaugh, S.E, (1998). *Computer Vision and Image Processing – a practical approach using CVIPtools*, Prentice Hall

[3] Burrus, C.S, Gopinath, R.A, Guo, H. (1998), *Introduction to Wavelets and Wavelets Transforms – A Primer*. New Jersey: Prentice Hall.

[4] Goswami, J, Chan A.K. (1999). *Fundamentals of Wavelets – Theory, Algorithms and Applications*. John Wiley

[5] Mallat S.G, A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-692, 1989.

Performance of Wavelet-based Image Compression on Syntbetic Images

Vooi Voon Yap[1], Shahedur Rahman[2]

[1]*Vision & Image Processing Group, School of Computing Science*
Middlesex University
Bounds Green Road, London N11 2NQ,, United KingdomTel: +44-02084116727,
E-mail:v.yap@mdx.ac.uk

[2]*Vision & Image Processing Group, School of Computing Science*
Middlesex University
Trent Park, Bramley Road, London N14 4YZ, UK, United Kingdom
Tel.: +44-208 411 6860, E-mail: s.rahman@mdx.ac.uk

Abstract

The main concern of this study is to identify the 'right' or 'most appropriate' wavelet for compressing still-synthetic images. To investigate this, different wavelets were used for a selected set of synthetic image. The results of this investigation are presented in this paper. The initial results show that different results can be obtained by using certain wavelets to compress still-synthetic images

Keywords

Wavelet, synthetic images, compression

1. Introduction

Since the advent of the Internet there has been an insatiable desire for faster transmission speed of still and video images using public telephone lines. A PAL colour video has 25 frames per second, 576 scan lines per frame, 720 pixels (pels) per scan lines for each of the red, green, and blue colour components. If each colour component is coded using 8 bits (24 bits/pixel total) then the bit-rate is 576 x 720 x 25 x 24 = 248 832 000 bits/sec or approximately 248.83 Mbits/sec [1]. The transmission times and storage capacity for uncompressed images are shown in Table 1 and Table 2 [2]. It is clear from the tables that transmitting such large amount of bits using normal communications link is very slow and the storage capacity is very inefficient. Thus, effective data compression techniques are essential for transmitting and storing digital images.

Table 1 Transmission time for uncompressed PAL video image

Duration of PAL video	File size (Kbits)	Transmission Time (hour)	
		Modem (56 Kbps)	E1 (2.048 Mbps)
1 sec	248 832	1.23	0.03
2 min	29859840	148.11	4.05
5 min	74649600	370.28	10.13

Table 2 Digitol medio storage capacities for PAL video imoge

Storage Media	Maximum storage capacity	
	No of Frames	Duration of video (min)
Magneto-Optical (MO)-640 Mbytes	515	0.34
MO – 2.6 Gbytes	2090	1.39
Recordable-CD 680 Mbytes	547	0.36
DVD – 8.5 Gbytes	6833	4.56
DVD – 17 Gbytes	13666	9.1

2. Synthetic Images

Images can be classified into different types and some researchers have attempted to classify images [3, 4] using different criteria. Smith and Chang have classified these different types of images as follows:

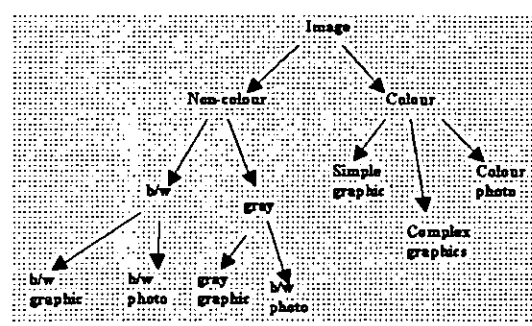


Figure 1 Image Classificotion

However, this method of classification is too elaborate for the purpose of this research. For this

research we have classified the images as follows:

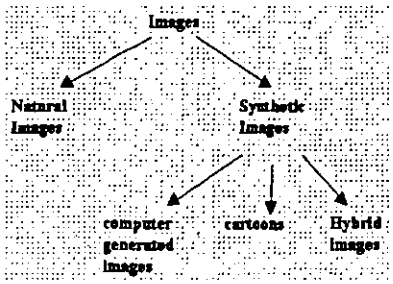


Figure 2 Synthetic Images Classification

The term 'synthetic image' referred to in this paper is an image that is generated by the computer or a drawing (cartoon). Hybrid image refers to an image that consists of a computer image superimposed on to a natural scene e.g. Walt Disney 'Dinosaur'. In this research, hybrid is also classified under synthetic images. It should also be noted that 3-D graphics like those used in engineering are included in the definition. Synthetic images have sharp artificial colour transitions whereas natural images have gradual colour transition. In a synthetic image sharp transition occurs between two different regions of constant colours. In contrast, natural images e.g. photographs do not have sharp edges like synthetic images instead the edges are often blurred. Synthetic images have fewer colours than natural images.

The transmission of synthetic images is very common and important on the Internet. Their impact is so significant that compression standard like MPEG-4 now supports synthetic visual objects such as human faces and body animation [5]. Previous MPEG standards only supported natural images. Virtual newscaster, ANANOVA is an example of synthetic images used on the Internet. For many engineering applications 3-D models are increasingly being accessed through the Internet [6]. Hence, it is only natural to conclude that synthetic objects and images will be playing more significant roles on the Internet in the future and therefore compression of synthetic images warrants further investigation.

3. Overview of DCT-based and Wavelet-based Compression

This section will give a brief overview of the Discrete Cosine Transform (DCT) based and wavelet-based compression methods.

3.1 JPEG

JPEG is a DCT-based compression standard for continuous-tone images. It is the most commonly used standard on the Internet. JPEG works on either full-colour or grey-scale images; it does not handle bi-level images well. It works best on 'continuous tone' or natural images but images like synthetic images with many sudden jumps in colour values will not compress well using JPEG. Nevertheless a lot of the synthetic images used on the Internet use JPEG compression.

The following is a brief overview of JPEG [5]. In JPEG the original image is taken through a series of steps, which are image preparation, DCT, quantization, and encoding. Each step contributes to the overall compression of the image.

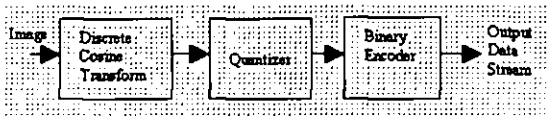


Figure 3 JPEG Compression

The JPEG standard uses the discrete cosine transform (DCT) and it suffers from annoying blocking artefacts when compression ratio is high. Increasing the compression ratio lowers the transmission time, but the image quality will deteriorate rapidly. The blocking artefacts were addressed in the new JPEG 2000 standard. JPEG 2000 uses wavelet transform rather than DCT. The advantage of wavelet transform is that, in contrast to JPEG, it does not divide the image into blocks, but analyse the image as a whole. Unlike the DCT used in the traditional JPEG, wavelet transform allows the high compression ratio and yet maintains the image quality [7].

3.2 Wavelet-based Compression

A typical wavelet-based image coder will comprise of three major parts: a wavelet filter bank, a quantizer, and an entropy coder.

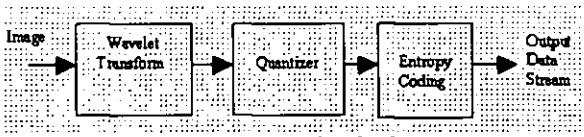


Figure 4 Wavelet Coder

The wavelet filter bank decomposes the image into wavelet coefficients. The quantizer then quantizes the wavelet coefficients. The entropy coder produces an output bit stream and then encodes these wavelet coefficients.

Although the overall performance of the coder depends on all three parts of the coder, the choice of the wavelet filter will ultimately affect the performance of the coder. If the wavelet filter performance is poor then however well the quantizer and encoder perform, it will not maintain the picture quality. Suffice to say filter banks play an important role in a typical wavelet-based coder. The choice of filters used in a number of researches is chosen on a 'trial-and-

error' basis. To date there are no known or published rules or guidelines to facilitate the selection of a wavelet filter for improved/optimised performance.

4. Experimental Method and Results

4.1 Compression Algorithm

The wavelet image coder used is written using Matlab. The algorithm [8] is outlined as follows:

Step 1: Perform a wavelet transform of the signal

Step 2: Set all values of the wavelet transform, which lie below some threshold value to 0.

Step 3: Transmit only non-zero values of the transform obtained from step 2.

Step 4: At the receiving end, perform an inverse wavelet transform of the data received from step 3.

The Matlab Wavelet toolbox function 'wavedec2' is used to perform wavelet transform. The image is decomposed into its coefficients using the 'wavedec2' function. The decomposition depends on the type of wavelet and the level of decomposition.

4.2 Wavelet Filters Used

There are a number of wavelets developed over the years. In this experiment only twelve are selected and they are: haar, db2, db5, db10, bior1.3, bior3.1, bior6.8, coif1, coif4, sym2, sym4, and sym8. The choice of wavelet filters are selected randomly for this research.

4.3 Images Used

For this particular study, a set of synthetic test images are put together from movies, and websites. Examples of these synthetic images can be found in the appendix.

4.4 Results and Discussions

Figure 5 shows the results of using different wavelets to compress different types of synthetic images. Three levels of decomposition are used. The image is decompose into 4*n subband images, where n is the number of level of decomposition. Using different wavelets, the performance of the image coder is evaluated based on the peak signal-to-noise ratio (PSNR) obtained. The PSNR is calculated by using the following formula:

$$PSNR = 10 * \log_{10}\left(255^2 / sqrt(MSE)\right)$$

and the mean square error (MSE) is obtained by:

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y) - I'(x,y)\right]^2$$

where M and N are the row and column of the image respectively. I is the original image and I' is the compressed image.

A high value of PSNR is good because it means that the signal-to-noise ratio is high. In image compression, the 'signal' is the original image, and the 'noise' is the error in reconstruction. The tabulated results are shown in the appendix. From the results (Figure 5), it can be seen that different wavelet filter produces different PSNR and the PSNR values vary from 75.23 db to 90.12 db.

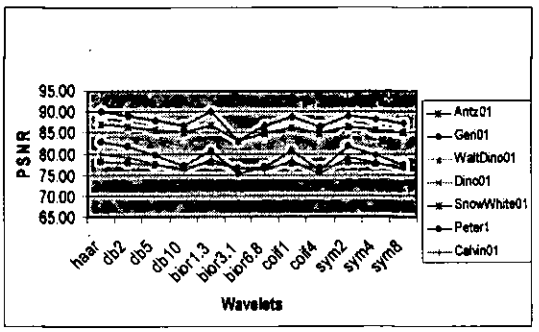


Figure 5 PSNR values for different wavelets

The results are better viewed in terms of the way the images have been categorized in this research.

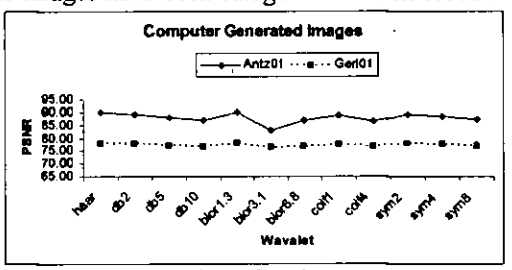


Figure 6 PSNR values for Computer Generated Images

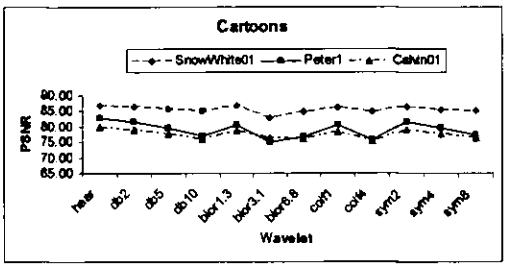


Figure 6 PSNR values for Cartoons

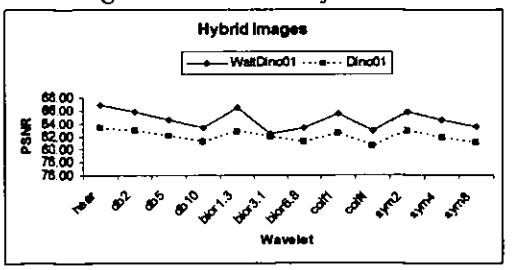


Figure 6 PSNR values for Hybrid Images

For each category of images, the PSNR varies widely, for example the computer generated image category the PSNR varies from 90.12 db to 76.45 db. This large variation in PSNR could be attributed to the characteristics of the image [9].

5. Conclusions and Future Work

One thing which is clear from the results obtained is that different wavelets produce different results. From Table 3 in the appendix, it can be seen that some wavelets filters performed better than others e.g. for Antz01 (computer generated image), the 'haar' wavelet filter performed best. And on the average the 'haar' performed the best. It is also observed that there is no specific wavelet filter that has performed consistently better than others. The results seem to indicate that PSNR values are image dependent i.e. the PSNR values depend on the type of image used.

The images from different categories used in this research have different characteristics. For example, cartoon images have fewer colours than the computer generated images and hybrid images. Cartoon images are basically line drawings, so their edges are more distinct than the edges found in the images from the other two categories.

As part of the research, a systematic study will be conducted to see if it is possible to establish a clear link between the characteristics of an image and the coding performance. It is hoped that the results from this study can help to design the 'right' or 'best' wavelet to compress still synthetic images.

To begin with, a systematic study has to be carried out to see if it is possible to establish a clear link between the characteristics of an image and the coding performance. A detail study will also be carried out to investigate the variables involved when deciding on a filter bank for a particular application. For example which wavelet filters type to use, e.g. orthorgonal or biorthogonal. The results from this study can then be used to help facilitate the design of the 'right' wavelet to compress synthetic images. It is envisaged that the results can also be extended in facilitating the selection of the 'right' wavelet for other images like medical images, aerial images, scanned images and compound images. A DSP-based image coding system using wavelets will be developed. An evaluation of the system will be carried out with the synthetic images.

6. References

[1] Haskell, B. G, Puri A, and Netravali, A N (1999). **Digital Video: An Introduction to MPEG-2**. London: Kluwer Academic Publishers.

[2] Oh, J and Woolley, S. I. (1999), "Diagnostic Quality Test for Wavelet-Compressed Digital Angiogram Images", The University of Birmingham, UK

[3] Smith, J.R, Chang, S (1997), "Multi-stage Classification of Image from Features and Related Text", IBM T J Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532; Dept. of Electrical Engineering Columbia University, New York, NY 10027

[4] Gevers, T., et al (2000). "Classification of Images on Internet by Visual and Textual Information",Internet Imaging, SPIE, San Jose.

[5] ISO/IEC JTC1/SC29/WG11 N4030, "Overview of the MPEG-4 Standard", March 2001. Online. Google. http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm. View date: February 2002.

[6] Rossignac, J., Safonova, A., Szymczak, A. (May 2001). "3D Compression Made Simple: Edgebreaker on a Corner-Table", Invited Lecture at the Shape Modeling International Conference, Gemoa, Italy.

[7] Pankaj, N. T. (1998). **Wavelet Image and Video Compression**. Boston: Kluwer Academic Publishers.

[8] Walker J, (1996). **A Primer on Wavelets and Their Scientific Applications**, CRC Press, Boca Raton.

[9] Saha, S., Vermur, R. (1999), "Adapative Wavelet Filters in Image Coders – How Important Are They?", Proc. IEEE IECON '99, San Jose, California.

7. Appendix

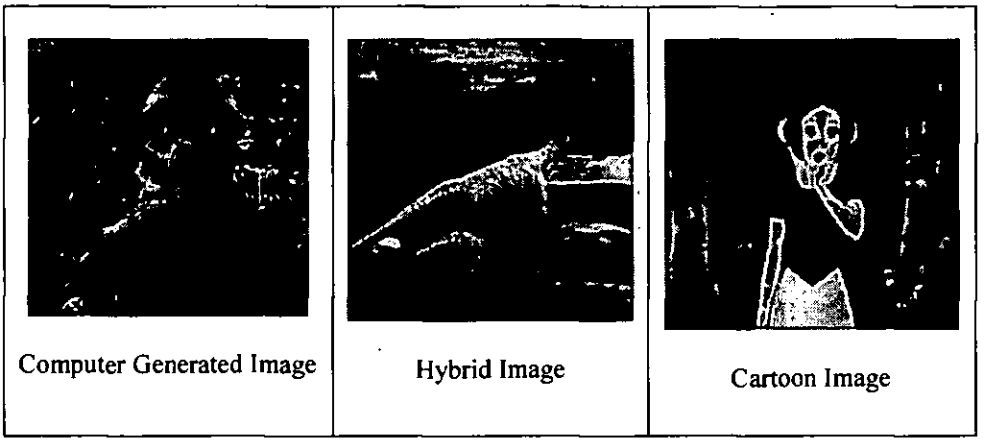


Figure7 Sample Images

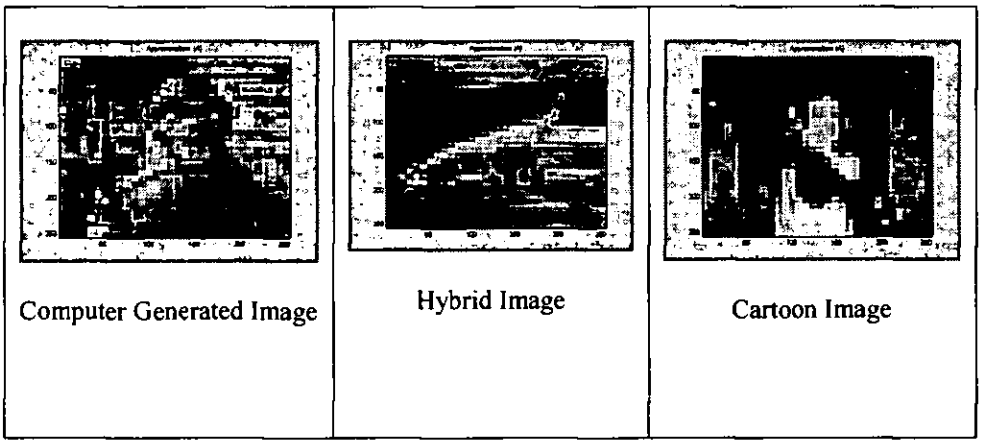


Figure 8 Images after 'haar' transform

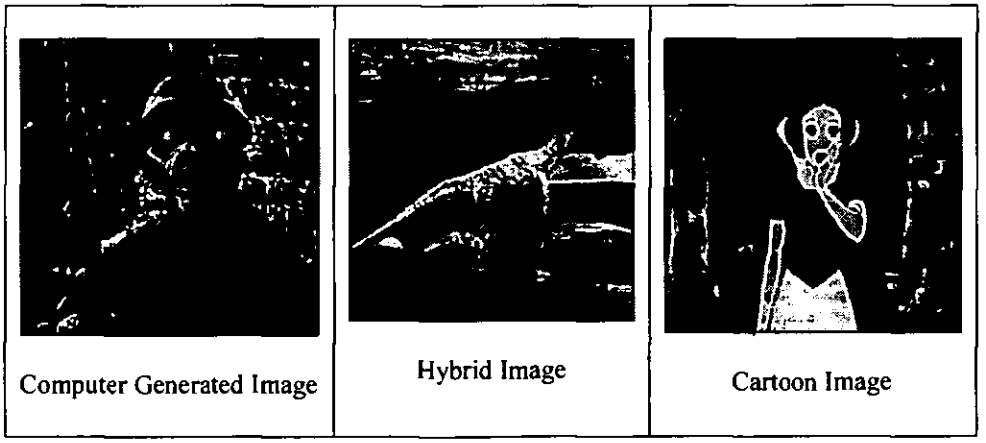


Figure 9 Images after reconstruction

Table 3 PSNR Values

Image	haar	db2	db5	db10	bior1.3	bior3.1	bior6.8	coif1	coif4	sym2	sym4	sym8
Antz01	90.12	89.08	87.98	86.75	90.07	83.10	86.69	88.78	86.42	89.08	88.18	87.17
Geri01	77.97	77.79	77.25	76.63	77.68	76.45	76.82	77.58	76.56	77.79	77.41	77.01
WaltDino01	86.98	85.82	84.51	83.25	86.43	82.37	83.30	85.45	82.87	85.82	84.64	83.54
Dino01	83.31	82.85	82.07	81.19	82.69	81.99	81.21	82.33	80.52	82.85	81.82	80.96
SnowWhite01	86.80	86.34	85.65	85.11	86.61	82.93	84.85	86.15	84.82	86.34	85.57	84.98
Peter1	82.80	81.67	79.53	76.98	80.68	75.23	76.83	80.67	75.86	81.67	79.63	77.29
Calvin01	79.82	78.97	77.57	75.99	78.53	76.42	75.94	78.38	75.26	78.97	77.76	76.25
Average	**83.97**	**83.22**	**82.08**	**80.84**	**83.24**	**79.78**	**80.81**	**82.76**	**80.33**	**83.22**	**82.14**	**81.03**

The Implications of Image Statistics and Image Features on Coding Performance of Synthetic Images

Vooi Voon Yap
Vision & Image Processing Group, School of Computing Science
Middlesex University
Bounds Green Road, London N11 2NQ, United Kingdom
Tel: +44-208 411 6727, E-mail:v.yap@mdx.ac.uk

Shahedur Rahman
Vision & Image Processing Group, School of Computing Science
Middlesex University
Trent Park, Bramley Road, London N14 4YZ, United Kingdom
Tel.: +44-208 411 6860, E-mail: s.rahman@mdx.ac.uk

ABSTRACT

The main emphasis of this research is to investigate which synthetic image characteristics affect the coding performance. The results indicate that coding performance is dependent on certain image statistics like edges, image gradient, skewness and kurtosis.

Keywords: Synthetic images, coding performance, wavelet, image statistics

I. SYNTHETIC IMAGES

The term 'synthetic image' refers to computer graphic image (CGI), a drawing (cartoon) or hybrid image. Hybrid image refers to an image that consists of a computer image superimposed on to a natural scene, e.g., Walt Disney 'Dinosaur'. Synthetic images have sharp artificial colour transitions whereas natural images have gradual colour transition. In a synthetic image sharp transition occurs between two different regions of constant colours. In contrast, natural images e.g. photographs do not have sharp edges like synthetic images instead the edges are often blurred. Synthetic images have fewer colours than natural images.

The transmission of synthetic images on the Internet is very common and important. Their impact is so significant that compression standard like MPEG-4 now supports synthetic visual objects such as human faces and body animation [5]. Hence, it is only natural to conclude that synthetic objects and images will be playing more significant roles on the Internet in the future and therefore compression of synthetic images warrants further investigation.

A typical wavelet-based image coder will comprise of three major parts: a wavelet filter bank, a quantizer, and an entropy coder.

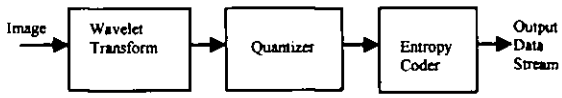


Figure 1 Wavelet Coder

In general, the operation of the image coder can be represented by the following notation;

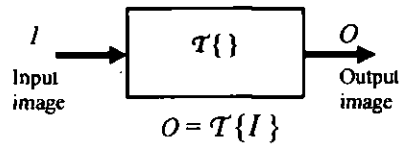


Figure 2 General representation of image coder

The above suggests that the output image, 'O' is related to the input image, 'I' by the transfer function 'T'.

From our previous work [1], we demonstrated that when compressing different synthetic images using different wavelet filter the peak signal-to-noise ratio (PSNR) is image dependent i.e., the PSNR varies from image to image. The result of that study is shown here in Figure 3.

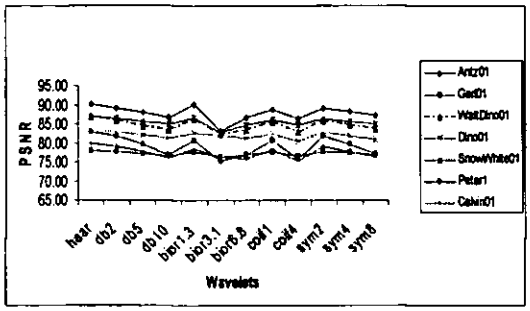


Figure 3 PSNR values for different wavelets

2. IMAGE STATISTICS

To further investigate why the PSNR varies from image to image, we analyse the images using gray-level image statistics [3] and image activity measure [6]. The image statistics used here are mean, standard deviation, variance, energy, entropy, skewness, and kurtosis.

2.1. Mean

$$Mean = \Sigma\, I(x,y)/(Nx * Ny) \quad (1)$$

where $\Sigma\, I(x,y)$ represents the summation of all pixel values of the image and $(Nx * Ny)$ is the size of the image.

2.2. Standard deviation

The measure of the frequency distribution of pixel values of an image is known as the standard deviation of that image. The standard deviation (Std. Dev.) can be calculated as shown below:

$$Std\ Dev = \sqrt{\left(\Sigma\, I(x,y)^2 / (Nx * Ny)\right) - \left(Mean^2\right)} \quad (2)$$

where $\Sigma\, I(x,y)^2$ is the sum of the squares of all pixel values of the image.

2.3. Variance

The variance (σ) is the square of the standard deviation and is calculated using the following:

$$Variance = (Std\ Dev)^2 \quad (3)$$

2.4. Energy

This refers to refers to how the grey-levels are being distributed [7] and is given by

$$E = \frac{1}{Nx * Ny} \Sigma\, I(x,y)^2 \quad (4)$$

where $Nx*Ny$ is the size of the image, and $I(x,y)$ is the value of the pixel of the image.

2.5. Entropy

Entropy is the measure of information content [4]. The entropy is measured by:

2.5.1 Normal entropy,

$$E_n = \Sigma\, I(x,y) \log I(x,y) \quad (5)$$

2.5.2 Shannon entropy,

$$E_s = -\frac{1}{Nx * Ny} \Sigma\, I(x,y)^2 \log\left(I(x,y)^2\right) \quad (6)$$

2.5.3 Log energy,

$$E_l = -\frac{1}{N} \Sigma \log\left(I(x,y)^2\right) \quad (7)$$

2.6. Skewness

Skewness is a measure of symmetry, or more precisely, the lack of symmetry of a histogram. A distribution, or data set is symmetric if it looks the same to the left and right of the centre point.

$$s = -\frac{1}{Nx * Ny} \Sigma \left(\frac{I(x,y) - mean}{\sigma}\right)^3 \quad (8)$$

2.7. Kurtosis

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. Data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails.

$$k = -\frac{1}{Nx * Ny} \Sigma \left(\frac{I(x,y) - mean}{\sigma}\right)^4 - 3 \quad (9)$$

3. IMAGE ACTIVITY MEASURE

In addition to the above-mentioned statistics features, image activity measure (IAM) is also used. IAM establishes how busy the image is in terms of edges and contours [6]. The following IAMs were used in this research i.e. edge information, and image gradient.

3.1 Edge Information

$$EI = \left(\frac{1}{Nx * Ny} \Sigma\, B(i)\right) * 100 \quad (10)$$

where $B(i)$ is the binary image derive from the Sobel edge extraction operator.

3.2 Image Gradient

This method calculates activity values by applying functions like the logarithm or square root to the horizontal and vertical gradient.

$$IG = \frac{1}{M*N}\left[\sum_{i=1}^{M-1}\sum_{j=1}^{N}|I(i,j)-I(i+1,j)| - \sum_{i=1}^{M}\sum_{j=1}^{N-1}|I(i,j)-I(i,j+1)|\right] \quad (11)$$

4. RESULTS

Various image statistics and image activity measure for a set of seven different synthetic images were calculated. The following tables show the results of the image statistics and image activity measure. Due to the restriction of space, the results are displayed as three separate tables.

Table 1 Image statistics

Image	Mean	Variance	Std Dev	Energy
Antz01	67.92	962.39	31.02	0.09
Geri01	117.97	5706.75	75.54	0.30
WaltDino01	93.69	1609.69	40.12	0.16
Dino01	134.89	1694.28	41.16	0.31
SnowWhite01	90.22	2337.96	48.35	0.16
Peter1	171.59	3040.53	55.14	0.50
Calvin01	207.44	2035.93	45.12	0.69

(a) Mean, variance, standard deviation, and energy

Image	Entropy			Skewness	Kurtosis
	Normal	Shannon	Log Energy		
Antz01	293.68	-50027	8.17	0.77	4.24
Geri01	591.67	-203746	8.54	0.1	1.75
WaltDino01	434.12	-99232	8.87	0.47	2.98
Dino01	668.49	-199942	9.68	-0.38	2.97
SnowWhite01	418.43	-101847	8.72	0.9	2.99
Peter1	892.99	-342806	10.14	-0.6	2.64
Calvin01	1112.2	-486600	10.61	-1.1	3.01

(b)Entropy, skewness and kurtosis

Table 2 Image activity measure

Image	Edge	Img Gradient
Antz01	3.905	13.957
Geri01	3.225	20.021
WaltDino01	3.128	15.402
Dino01	2.002	7.627
SnowWhite01	2.982	5.705
Peter1	5.414	50.571
Calvin01	5.513	18.876

Different wavelets are used to compress different types of synthetic images. Three levels of decomposition are used. The image is decomposed into 4*n subband images, where n is the number of level of decomposition. Due to the limitation of space, we are presenting the results for one wavelet (Haar) only.

Table 3 Performance of image coder

Image	Haar		
	PSNR	Comp. Ratio	bpp
Antz01	90.12	4.45	5.40
Geri01	77.97	4.31	5.56
WaltDino01	86.97	6.22	3.85
Dino01	83.31	19.09	1.25
SnowWhite01	86.80	7.30	3.29
Peter1	82.79	3.21	7.45
Calvin01	79.81	5.68	4.22

The following coding performance measures were found: peak signal-to-noise ratio (PSNR), compression ratio (CR), and bits-per-pixel (bpp). Again due to the limitation of space, only samples of results are presented here (Figure 4 - 8).

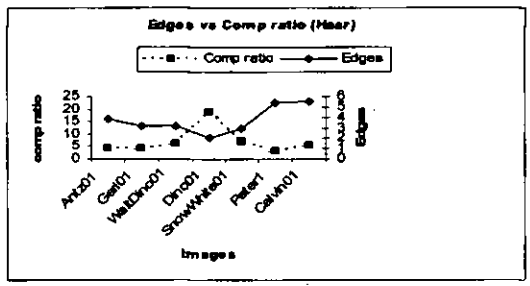


Figure 4 Edges versus compression ratio values

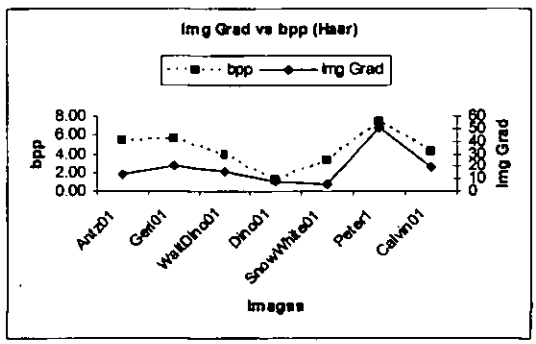


Figure 5 Image gradient versus bpp values

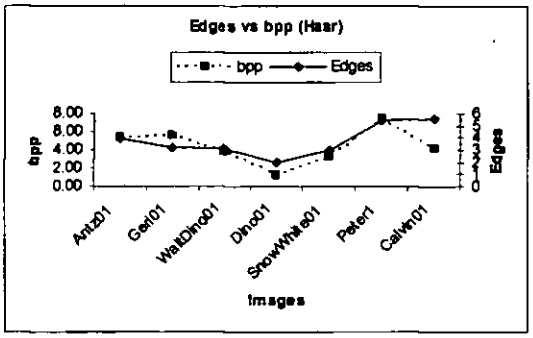


Figure 6 Edges versus bpp values

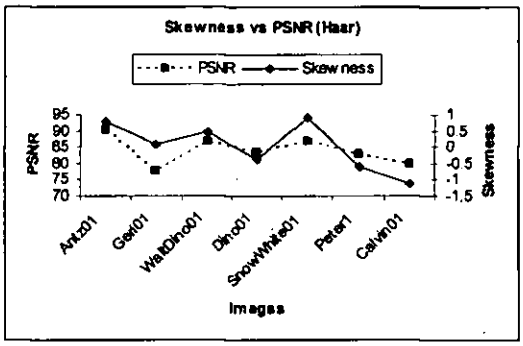


Figure 7 Skewness versus PSNR values

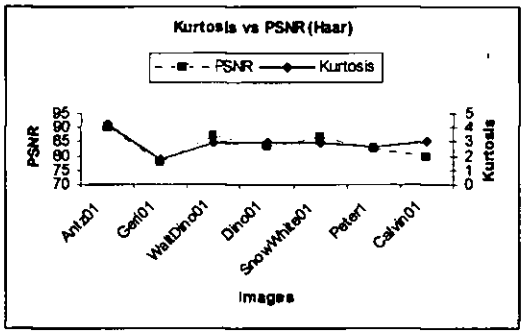


Figure 8 Kurtosis versus PSNR values

The results from analysis show that a negative correlation exists between the edges and compression ratio. The analysis also reveals that the image gradient and bpp are positively correlated. A positive correlation also exists between the edges and bpp. The skewness values show a strong positive correlation with PSNR values. A strong positive correlation also exists between the kurtosis and PSNR values.

5. CONCULSION AND FUTURE WORK

A systematic study has been conducted to see if there is any link between the characteristics of an image and the coding performance.

The results obtained show that different wavelets produce different results. It is also observed that there is no specific wavelet filter that has performed consistently better than others. Although the study found no clear relation between most image statistics and the coding performance measures, however the results does indicate that coding performance measures is related to certain image statistics like edges, image gradient, skewness and kurtosis.

The different images used in this research have different characteristics. For example, cartoon images have fewer colours than the computer generated images and hybrid images. Cartoon images are basically line drawings, so their edges are more distinct than the edges found in CGI and hybrid image. The analysis of these images has shown that kurtosis has the strongest correlation with the PSNR.

This study also exposed the issues that have yet to be resolved pertaining to the choice of wavelets for compressing synthetic images. For examples, is it the limited number of colours, is it the brightness, or is it the distinct edges of a synthetic image that influence the coding performance?

It is envisaged that the results can help to design the right wavelet filters to be used for compressing synthetic images for better performance. It is the researchers' opinion that using the 'right' wavelet filters is very important. The reason is that if the performance of the wavelet filter is poor than the overall performance of the wavelet coder cannot be significant improved no matter how well the quantization and entropy encoder perform.

It is also envisaged that the results can also be extended in facilitating the selection of the 'right' wavelet for other images like medical images, aerial images, scanned images and compound images.

6. REFERENCES

[1] Vooi Voon, Yap, Shahedur, Rahman, "Performance of Wavelet-based Image Compression on Synthetic Images", **International Conference on Robotics, Vision, Information And Signal Processing**, IEEE Malaysia, Penang, Malaysia , January 2003

[2] Subhasis, Saha, Vemuri, Rao, "Adaptive Wavelets Filters in Image Coders – How Important Are They?", **Proc. IEEE IECON '99, 1999**

[3] Subhasis, Saha, Vemuri, Rao, "How Do Image Statistics Imapct Lossy Coding

Performance?", **Proc. IEEE International Confereoce on Information Technology: Coding and Computing (ITCC)**, 2002.

[4] Parker, J.R., **Algorithms for Image Processing and Computer Vision**, Wiley, USA, 1997

[5] ISO/IEC JTC1/SC29/WG11 N4030, "Overview of the MPEG-4 Standard", March 2001. Online. Google. http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm. View date: February 2002.

[6] Subhasis, Saha, Vemuri, Rao, "An Analysis on the Effect of Image Features on Lossy Coding Performance", **IEEE Sigoal Processing Letters**, May, 2002.

[7] Umbaugh, S.E, **Computer Vision and Image Processing – a practical approach using CVIPtools**, Prentice Hall, 1988.

APPENDIX 4

This appendix contains the Matlab source code for the main function used in this research.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author : Vooi Voon YAP, Middlesex University, London                                        %
%---------------------------------------------------------------------------------------------%
% First written: 20/03/2002                                                                   %
% Last Update:   25/03/2004                                                                   %
% MSE calculations revised in this version. - 03/01/2004                                      %
% Corrected 'dequantize' error - used dequantiz(decodedCr,Cbscale) instead of                 %
% dequantiz(decodedCr,Crscale) for dequantizedCr.                                             %
%---------------------------------------------------------------------------------------------%
% Main Function : ColorWavCompressV3a.m                                                       %
%---------------------------------------------------------------------------------------------%
% Purpose :                                                                                   %
% This is a wavelet-based image compression program written for my PhD research.              %
% It compress colour images using DWT.                                                        %
%---------------------------------------------------------------------------------------------%
% Note: This function/program calls in functions written by fellow researchers:               %
% This function threshold the YCbCr coefficients separately.                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

% Orig_Img=imread('D:\matlab6p5\work\images\natural\8mile01_256.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\aerial\aerial01_256.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\peppers.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\lenacolor.bmp');
Orig_Img=imread('D:\matlab6p5\work\images\natural\palmblades.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\boy.bmp');

% Orig_Img=imread('D:\matlab6p5\work\images\natural\bird.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\marcie.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\chiliq01.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\mandrill.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\chiliq02.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\flower256.bmp');


figure(1)
imshow(Orig_Img); title('Original Image');  % display image
Orig_Img = double(Orig_Img);                % convert to double for arithmetic operations

ROrig=Orig_Img(:,:,1);
GOrig=Orig_Img(:,:,2);
BOrig=Orig_Img(:,:,3);

%%%%%%%%%%%%%%%%%%%%%%%%%%% Transform to ICT colour space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y  = 0.299*Orig_Img(:,:,1) + 0.587*Orig_Img(:,:,2) + 0.114*Orig_Img(:,:,3);
Cb = -0.16875*Orig_Img(:,:,1) - 0.33126*Orig_Img(:,:,2) + 0.5*Orig_Img(:,:,3);
Cr = 0.5*Orig_Img(:,:,1) - 0.41869*Orig_Img(:,:,2) - 0.08131*Orig_Img(:,:,3);

Final_Img = zeros(size(Orig_Img));          % Clear final image matrix (reset)

dwtmode('sym');

YLevel  = 3;                              % level of the decomposition
CbCrLevel = 3;
WaveletType1 ='haar';                       % name of the wavelet
WaveletType2 ='haar';

[YRecon, YcomEncoded, YcomTotalCoeff, YcomNonZeroCoeff]...
    = YComp(Y, WaveletType1, YLevel);
Y = YRecon;
[CbRecon, CbcomEncoded, CbcomTotalCoeff, CbcomNonZeroCoeff]...
    = CbComp(Cb, WaveletType2, CbCrLevel);
Cb = CbRecon;
[CrRecon, CrcomEncoded, CrcomTotalCoeff, CrcomNonZeroCoeff]...
    = CrComp(Cr, WaveletType2, CbCrLevel);
Cr = CrRecon;

%%%%%%%%%%%%%%%%%% Perform inverse transform to reconstruct the RGB image %%%%%%%%%%%%%%%
R = Y + 1.402.*Cr;
G = Y + (-0.34413.*Cb) + (-0.71414.*Cr);
B = Y + 1.772.*Cb ;
```

```
Final_Img(:,:,1)=R;
Final_Img(:,:,2)=G;
Final_Img(:,:,3)=B;

CmpR=Final_Img(:,:,1);
CmpG=Final_Img(:,:,2);
CmpB=Final_Img(:,:,3);

Final_Img=uint8(Final_Img);
figure(2)
imshow(Final_Img);title('Reconstructed Image');

imwrite(Final_Img,'D:\Matlab6p5\work\Images\DummyImg.bmp');
fprintf (1, 'YcomEncoded:                %d\n', size(YcomEncoded,1));
fprintf (1, 'CbcomEncoded:               %d\n', size(CbcomEncoded,1));
fprintf (1, 'CrcomEncoded:               %d\n', size(CrcomEncoded,1));

Rerror = ROrig - CmpR;
[Rm1 Rn1] = size(ROrig);
RMSE = (sum(sum(Rerror.*Rerror)))./(Rm1*Rn1);
Gerror = GOrig - CmpG;
[Gm1 Gn1] = size(GOrig);
GMSE = (sum(sum(Gerror.*Gerror)))./(Gm1*Gn1);
Berror = BOrig - CmpB;
[Bm1 Bn1] = size(BOrig);
BMSE = (sum(sum(Berror.*Berror)))./(Bm1*Bn1);

fprintf (1, 'Y Decomposition level1:       %d\n', YLevel);
fprintf (1, 'CbCr Decomposition level2:    %d\n', CbCrLevel);

fprintf (1, 'Y Wavelet Type:               %s\n', WaveletType1);
fprintf (1, 'CbCr Wavelet Type:            %s\n', WaveletType2);


RGBMSE=RMSE+GMSE+BMSE;
AveMSE=RGBMSE/3;
fprintf(1, '\nAverage MSE is:                      %1.4f', AveMSE);

RGBPSNR = 0;
RGBPSNR = 10*log10((255*255)/AveMSE);
fprintf(1, '\nRGB PSNR is:                        %1.3f dB\n', RGBPSNR);

TotalCoeff(1) = YcomTotalCoeff;
TotalCoeff(2) = CbcomTotalCoeff;
TotalCoeff(3) = CrcomTotalCoeff;

NonZeroCoeff(1) = YcomNonZeroCoeff;
NonZeroCoeff(2) = CbcomNonZeroCoeff;
NonZeroCoeff(3) = CrcomNonZeroCoeff;

CoeffCompRatio = sum(TotalCoeff)/sum(NonZeroCoeff);
fprintf (1, 'Coeff. Compression ratio:           %2.3f\n', CoeffCompRatio);

ImgSz=(size(Orig_Img,1)*size(Orig_Img,2))*3;
CompressedImage=size(YcomEncoded,1)+size(CbcomEncoded,1)+size(CrcomEncoded,1);
FileCompRatio = ImgSz/CompressedImage;'
fprintf (1, 'File Compression ratio:             %2.3f\n', FileCompRatio);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% bits per pixel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bpp = (8*CompressedImage)/ImgSz;
fprintf(1, 'bpp :                                %1.3f\n\n', bpp);

Totalencoded = size(YcomEncoded,1)+size(CbcomEncoded,1)+size(CrcomEncoded,1);
fprintf(1, '\nTotal encoded output :       %d', Totalencoded);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author : Vooi Voon YAP, Middlesex University, London                                     %
%------------------------------------------------------------------------------------------%
% First written: 14/03/2004                                                                 %
% Last Update:   26/03/2004                                                                 %
%------------------------------------------------------------------------------------------%
% Function : YComp.m                                                                        %
%------------------------------------------------------------------------------------------%
% Purpose :                                                                                 %
% This is a wavelet-based image compression program written for my PhD research.           %
% It process gray-scale images.                                                             %
%------------------------------------------------------------------------------------------%
% Note: This function/program calls in functions written by other fellow researchers:      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [oImage, yEncoded, yTotalCoeff, yNonZeroCoeff]...
    = YComp(Yimage, WaveletName, Level)

Orig_Img = Yimage; % re-assign image

Orig_Img = double(Orig_Img);                % convert to double for arithmetic operations

Final_Img = zeros(size(Orig_Img));         % Clear final image matrix (reset)

dwtmode('sym');

DecomLevel = Level;                          % level of the decomposition
WaveletType = WaveletName;                    % name of the wavelet

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------- Compress each component of the image individually --------------%

[Basis, l] = wavedec2(Orig_Img,DecomLevel,WaveletType);
TotalCoeff(1) = size(Basis,2);          % used for computing compression ratio

%---------------------------------------- Threshold --------------------------------------%
Tbasis1=wthresh(Basis,'h',20);     %20

%------------------------------------ quantization ----------------------------------------%
[quantized,scale] = quantiz(Tbasis1,8);  %16
NonZeroCoeff(1)=nnz(quantized);             % compute the no. quantized thresholded
coefficients left

%----------------------------------- entropy encoding -----------------------------------%
[ni,nj] = size(quantized);
encoded = EntropyCoder(quantized,'encode',ni,nj);
yEncoded = encoded;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------------------------------- decode ---------------------------------------%
decoded = EntropyCoder(encoded,'decode',ni,nj);

%-------------------------------------------- dequantize ---------------------------------%
dequantized = dequantiz(decoded,scale);

%--------------------------------------- reconstruction ----------------------------------%
ReconCoeff = waverec2(dequantized, l, WaveletType);
oImage  = ReconCoeff;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------------------------- Calculation of compression ratio -------------------------%
yTotalCoeff = TotalCoeff;
yNonZeroCoeff = NonZeroCoeff;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author : Vooi Voon YAP, Middlesex University, London                                     %
%------------------------------------------------------------------------------------------%
% First written: 14/03/2004                                                                %
% Last Update:   26/03/2004                                                                %
%------------------------------------------------------------------------------------------%
% Function : CbComp.m
%
%------------------------------------------------------------------------------------------%
% Purpose :                                                                                %
% This is a wavelet-based image compression program written for my PhD research.           %
% It process gray-scale images.                                                            %
%------------------------------------------------------------------------------------------%
% Note: This function/program calls in functions written by other fellow researchers:      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [oImage, CbEncoded, CbTotalCoeff, CbNonZeroCoeff]...
    = CbComp(Cbimage, WaveletName, Level)

Orig_Img = Cbimage; % re-assign image

Orig_Img = double(Orig_Img);                % convert to double for arithmetic operations

Final_Img = zeros(size(Orig_Img));        % Clear final image matrix (reset)

dwtmode('sym');

DecomLevel = Level;                           % level of the decomposition
WaveletType = WaveletName;                     % name of the wavelet

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%----------------------- Compress each component of the image individually ---------------%

[Basis, l] = wavedec2(Orig_Img,DecomLevel,WaveletType);
TotalCoeff(1) = size(Basis,2);          % used for computing compression ratio

%--------------------------------------- Threshold ----------------------------------------%
Tbasisl=wthresh(Basis,'h',20);      %20

%----------------------------------------- quantization ----------------------------------%
[quantized,scale] = quantiz(Tbasisl,8);  %8 max:16
NonZeroCoeff(l)=nnz(quantized);              % compute the no. quantized thresholded
coefficients left

%-------------------------------------- entropy encoding ---------------------------------%
[ni,nj] = size(quantized);
encoded = EntropyCoder(quantized,'encode',ni,nj);
CbEncoded = encoded;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------------------------------ decode ---------------------------------------%
decoded = EntropyCoder(encoded,'decode',ni,nj);

%--------------------------------------------- dequantize --------------------------------%
dequantized = dequantiz(decoded,scale);

%------------------------------------------- reconstruction ------------------------------%
ReconCoeff = waverec2(dequantized, l, WaveletType);
oImage  = ReconCoeff;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------------------------- Calculation of compression ratio ---------------------------%
CbTotalCoeff = TotalCoeff;
CbNonZeroCoeff = NonZeroCoeff;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author : Vooi Voon YAP, Middlesex University, London                                          %
%-----------------------------------------------------------------------------------------------%
% First written: 14/03/2004                                                                     %
% Last Update:   26/03/2004                                                                     %
%-----------------------------------------------------------------------------------------------%
% Function : CrComp.m                                                                           %
%-----------------------------------------------------------------------------------------------%
% Purpose :
%
% This is a wavelet-based image compression program written for my PhD research.
%
% It process gray-scale images.
%
%-----------------------------------------------------------------------------------------------%
% Note: This function/program calls in functions written by other fellow researchers:          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [oImage, CrEncoded, CrTotalCoeff, CrNonZeroCoeff]...
    = CrComp(Crimage, WaveletName, Level)

Orig_Img = Crimage; % re-assign image

Orig_Img = double(Orig_Img);                % convert to double for arithmetic operations

Final_Img = zeros(size(Orig_Img));        % Clear final image matrix (reset)

dwtmode('sym');

DecomLevel = Level;                          % level of the decomposition
WaveletType = WaveletName;                    % name of the wavelet

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------- Compress each component of the image individually ---------------%

[Basis, l] = wavedec2(Orig_Img,DecomLevel,WaveletType);
TotalCoeff(1) = size(Basis,2);         % used for computing compression ratio

%-------------------------------------------- Threshold ---------------------------------%
Tbasis1=wthresh(Basis,'h',20);     %20

%--------------------------------------- quantization -----------------------------------%
[quantized,scale] = quantiz(Tbasis1,8);  %8 max:16
NonZeroCoeff(1)=nnz(quantized);             % compute the no. quantized thresholded
coefficients left

%--------------------------------------- entropy encoding --------------------------------
---
[ni,nj] = size(quantized);
encoded = EntropyCoder(quantized,'encode',ni,nj);
CrEncoded = encoded;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of encoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% begin decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------------------ decode --------------------------
decoded = EntropyCoder(encoded,'decode',ni,nj);

%-------------------------- dequantize --------------------------
dequantized = dequantiz(decoded,scale);

%---------------------- reconstruction --------------------------
ReconCoeff = waverec2(dequantized, l, WaveletType);
oImage  = ReconCoeff;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of decoding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------------------------- Calculation of compression ratio -------------------------%
CrTotalCoeff = TotalCoeff;
CrNonZeroCoeff = NonZeroCoeff;
```

```
function out = EntropyCoder(in,state,ni,nj)
% Entropy Coder using Huffman encoding. The Huffman coding function called in this program
is
% a third party function created by Karl Skretting, (Hogskolen in Stavanger (Stavanger
% University)
% Signal Processing Group, karl.skretting@tn.his.no
% Homepage:  http://www.ux.his.no/~karlsk/)
%
% inputs:
% in - the input matrix
% state - can be the string 'encode' or 'decode' to specify encode or decode process
% ni & nj - dimensions of the input matrix
%
% output:
% out - the encoded image in the form of a sequence


if isequal(state,'encode')
    % Convert the matrix c into a 1-D array of coefficients
    cc = reshape(in,ni*nj,1);
    % Prepare input for the Huffman function
    xC=cell(2,1); xC{1}=cc;
    % Pass to the Huffman encoding function
    out = Huff06(xC);
    return;
elseif isequal(state,'decode')
    % Pass to Huffman decoding function
    xC = Huff06(in);
    % Convert back to matrix version
    out = reshape(xC{1},ni,nj);
    return;
else
    fprintf('Input state error, please specify the encode/decode process');
    return;
end



function varargout = Huff06(xC, ArgLevel, ArgSpeed)
% Huff06      Huffman encoder/decoder with (or without) recursive splitting
% Vectors of integers are Huffman encoded,
% these vectors are collected in a cell array, xC.
% If first argument is a cell array the function do encoding,
% else decoding is done.
%
% [y, Res] = Huff06(xC, Level, Speed);                    % encoding
% y = Huff06(xC);                                         % encoding
% xC = Huff06(y);                                         % decoding
% ------------------------------------------------------------------
% Arguments:
%  y        a column vector of non-negative integers (bytes) representing
%           the code, 0 <= y(i) <= 255.
%  Res      a matrix that sum up the results, size is (NumOfX+1)x4
%           one line for each of the input sequences, the columns are
%           Res(:,1) - number of elements in the sequence
%           Res(:,2) - zero-order entropy of the sequence
%           Res(:,3) - bits needed to code the sequence
%           Res(:,4) - bit rate for the sequence, Res(:,3)/Res(:,1)
%           Then the last line is total (which include bits needed to store NumOfX)
%  xC       a cell array of column vectors of integers representing the
%           symbol sequences. (should not be to large integers)
%           If only one sequence is to be coded, we must make the cell array
%           like: xC=cell(2,1); xC{1}=x; % where x is the sequence
%  Level    How many levels of splitting that is allowed, legal values 1-8
%           If Level=1, no further splitting of the sequences will be done
%           and there will be no recursive splitting.
%  Speed    For complete coding set Speed to 0. Set Speed to 1 to cheat,
%           during encoding, y will then be a sequence of zeros only,
%           but it will be of correct length and the other output
%           arguments will be correct.
% ------------------------------------------------------------------

% SOME NOTES ON THE FUNCTION
% huff06 depends on other functions for Huffman code, and the functions in this file
%  HuffLen      - find length of codewords (HL)
%  HuffTabLen   - find bits needed to store Huffman table information (HL)
%  HuffCode     - find huffman codewords
```

```
%  HuffTree    - find huffman tree

%-------------------------------------------------------------------------
% Copyright (c) 1999-2000.  Karl Skretting.  All rights reserved.
% Hogskolen in Stavanger (Stavanger University), Signal Processing Group
% Mail:  karl.skretting@tn.his.no   Homepage:  http://www:ux.his.no/~karlsk/
%
% HISTORY:
% Ver. 1.0  13.06.2000  KS: Function made based on huff04
% Ver. 1.1  20.06.2000  KS: Handle some more exceptions
% Ver. 1.2  21.06.2000  KS: Handle also negative values
% Ver. 1.3  23.06.2000  KS: Use logarithms for some sequences (line 114)
% Ver. 1.4  31.07.2000  KS: If a sequence has many zeros, Run + Value coding
%   is done. (from line 255 and some more)
% Ver. 1.5  02.08.2000  KS: May have larger integers in PutVLIC and GetVLIC
% Ver. 1.6  18.01.2001  KS: MaxL in line 218 was reduced from 2^16 to 50000.
%   For some sequences we may have length of code word larger than 16, even
%   if probability was larger than 2^(-16). Ex: Hi=[12798,14241,7126,7159,3520,...
%   3512,1857,1799,1089,1092,681,680,424,431,320,304,201,204,115,118,77,83,45,...
%   40,24,26,18,14,4,12,3,3,4,2,2,0,1]', sum(Hi)=58029
% Ver. 1.7  21.08.2001  KS: MaxL in line 218 and 420 must be the same
%   We may now have long code words (also see HuffTabLen.m)
%-------------------------------------------------------------------------

global y Byte BitPos Speed Level

Mfile='Huff06';
Debug=0;    % note Debug is defined in EncodeVector and DecodeVector too

% check input and output arguments, and assign values to arguments
if (nargin < 1);
   error([Mfile,': function must have input arguments, see help.']);
end
if (nargout < 1);
   error([Mfile,': function must have output arguments, see help.']);
end

if (~iscell(xC))
   Encode=0;Decode=1;
   y=xC(:);            % first argument is y
else
   Encode=1;Decode=0;
   if (nargin < 3); Speed=0; else Speed=ArgSpeed; end;
   if (nargin < 2); Level=8; else Level=ArgLevel; end;
   if ((length(Speed(:))~=1));
      error([Mfile,': Speed argument is not scalar, see help.']);
   end
   if Speed; Speed=1; end;
   if ((length(Level(:))~=1));
      error([Mfile,': Level argument is not scalar, see help.']);
   end
   Level=floor(Level);
   if (Level < 1); Level=1; end;
   if (Level > 8); Level=8; end;
   NumOfX = length(xC);
end

if Encode
   Res=zeros(NumOfX,4);
   % initalize the global variables
   y=zeros(10,1);    % put some zeros into y initially
   Byte=0;BitPos=1;  % ready to write into first position
   % start encoding, first write VLIC to give number of sequences
   PutVLIC(NumOfX);
   if Debug
      disp([Mfile,' (Encode): Level=',int2str(Level),'  Speed=',int2str(Speed),...
            '  NumOfX=',int2str(NumOfX)]);
   end
   % now encode each sequence continuously
   Ltot=0;
   for num=1:NumOfX
      x=xC{num};
      x=full(x(:));        % make sure x is a non-sparse column vector
      L=length(x);Ltot=Ltot+L;
      y=[y(1:Byte);zeros(50+2*L,1)];  % make more space available in y
      % now find some info about x to better code it
      maxx=max(x);
```

```
      minx=min(x);
      if (minx<0)
         Negative=1;
      else
         Negative=0;
      end
      if ( (((maxx*4)>L) | (maxx>1023)) & (L>1) & (maxx>minx))
         % the test for LogCode could be better, I think, (ver. 1.3)
         LogCode=1;    % this could be 0 if LogCode is not wanted
      else
         LogCode=0;
      end
      PutBit(LogCode);
      PutBit(Negative);
      I=find(x);                        % non-zero entries in x
      Sg=(sign(x(I))+1)/2;              % the signs may be needed later, 0/1
      x=abs(x);
      if LogCode
         xa=x;                          % additional bits
         x(I)=floor(log2(x(I)));
         xa(I)=xa(I)-2.^x(I);
         x(I)=x(I)+1;
      end
      [bits, ent]=EncodeVector(x);   % store the (abs and/or log) values
      if Negative                       % store the signs
         for i=1:length(Sg); PutBit(Sg(i)); end;
         bits=bits+length(Sg);
         ent=ent+length(Sg)/L;
      end
      if LogCode                        % store the additional bits
         for i=1:L
            for ii=(x(i)-1):(-1):1
               PutBit(bitget(xa(i),ii));
            end
         end
         bits=bits+sum(x)-length(I);
         ent=ent+(sum(x)-length(I))/L;
      end
      if L>0; Res(num,1)=L; else Res(num,1)=1; end;
      Res(num,2)=ent;
      Res(num,3)=bits;
   end
   y=y(1:Byte);
   varargout(1) = {y};
   if (nargout >= 2)
      % now calculate results for the total
      if Ltot<1; Ltot=1; end;   % we do not want Ltot to be zero
      Res(NumOfX+1,3)=Byte*8;
      Res(NumOfX+1,1)=Ltot;
      Res(NumOfX+1,2)=sum(Res(1:NumOfX,1).*Res(1:NumOfX,2))/Ltot;
      Res(:,4)=Res(:,3)./Res(:,1);
      varargout(2) = {Res};
   end
end

if Decode
   % initalize the global variables, y is set earlier
   Byte=0;BitPos=1;  % ready to read from first position
   NumOfX=GetVLIC;   % first read number of sequences
   if Debug
      disp([Mfile,'(Decode):  NumOfX=',int2str(NumOfX),'  length(y)=',int2str(length(y))]);
   end
   xC=cell(NumOfX,1);
   for num=1:NumOfX
      LogCode=GetBit;
      Negative=GetBit;
      x=DecodeVector;   % get the (abs and/or log) values
      L=length(x);
      I=find(x);
      if Negative
         Sg=zeros(size(I));
         for i=1:length(I); Sg(i)=GetBit; end;   % and the signs   (0/1)
         Sg=Sg*2-1;                              % (-1/1)
      else
         Sg=ones(size(I));
      end
      if LogCode          % read additional bits too
```

```
            xa=zeros(L,1);
            for i=1:L
               for ii=2:x(i)
                  xa(i)=2*xa(i)+GetBit;
               end
            end
            x(I)=2.^(x(I)-1);
            x=x+xa;
         end
         x(I)=x(I).*Sg;
         xC{num}=x;
      end
      varargout{1} = {xC};
end

return     % end of main function, huff06

% the EncodeVector and DecodeVector functions are the ones
% where actual coding is going on.
% This function calls itself recursively
function [bits, ent] = EncodeVector(x, bits, HL, Maxx, Meanx)
global y Byte BitPos Speed Level
Debug=0;
Level = Level - 1;
MaxL=50000;          % longer sequences is split in the middle
L=length(x);
% first handle some special possible exceptions,
if L==0
   PutBit(0);        % indicate that a sequence is coded
   PutVLIC(L);       % with length 0 (0 is 6 bits)
   PutBit(0);        % 'confirm' this by a '0', Run + Value is indicated by a '1'
   bits=2+6;
   ent=0;
   Level = Level + 1;
   return    % end of EncodeVector
end
if L==1
   PutBit(0);        % indicate that a sequence is coded
   PutVLIC(L);       % with length 1 (6 bits)
   PutVLIC(x(1));    % containing this integer
   bits=1+2*6;
   if (x(1)>=16); bits=bits+4; end;
   if (x(1)>=272); bits=bits+4; end;
   if (x(1)>=4368); bits=bits+5; end;
   if (x(1)>=69904); bits=bits+5; end;
   if (x(1)>=1118480); bits=bits+4; end;
   ent=0;
   Level = Level + 1;
   return    % end of EncodeVector
end
if max(x)==min(x)
   PutBit(0);        % indicate that a sequence is coded
   PutVLIC(L);       % with length L
   for i=1:7; PutBit(1); end;   % write end of Huffman Table
   PutVLIC(x(1));    % containing this integer
   bits=1+6+7+6;
   if (x(1)>=16); bits=bits+4; end;
   if (x(1)>=272); bits=bits+4; end;
   if (x(1)>=4368); bits=bits+5; end;
   if (x(1)>=69904); bits=bits+5; end;
   if (x(1)>=1118480); bits=bits+4; end;
   if (L>=16); bits=bits+4; end;
   if (L>=272); bits=bits+4; end;
   if (L>=4368); bits=bits+5; end;
   if (L>=69904); bits=bits+5; end;
   if (L>=1118480); bits=bits+4; end;
   ent=0;
   Level = Level + 1;
   return    % end of EncodeVector
end
% here we test if Run + Value coding should be done
I=find(x);   % the non-zero indices of x
if (L/2-length(I))>50
   Maxx=max(x);
   Hi=IntHist(x,0,Maxx);  % find the histogram
   Hinz=nonzeros(Hi);
   ent=log2(L)-sum(Hinz.*log2(Hinz))/L;  % find entropy
```

```
   % there are few non-zero indices => Run+Value coding of x
   x2=x(I);  % the values
   I=[I(:);L+1];   % include length of x
   for i=length(I):(-1):2; I(i)=I(i)-I(i-1); end;
   x1=I-1;   % the runs
   % code this as an unconditional split (like if L is large)
   if Speed
      Byte=Byte+1;    % since we add 8 bits
   else
      PutBit(0);       % this is idicated like when a sequence
      PutVLIC(0);      % of length 0 is coded, but we add one extra bit
      PutBit(1);       % Run + Value is indicated by a '1'
   end;
   [bits1, temp] = EncodeVector(x1);
   [bits2, temp] = EncodeVector(x2);
   bits=bits1+bits2+8;
   Level = Level + 1;
   return    % end of EncodeVector
end

if (nargin==1)
   Maxx=max(x);
   Meanx=mean(x);
   Hi=IntHist(x,0,Maxx);  % find the histogram
   Hinz=nonzeros(Hi);
   ent=log2(L)-sum(Hinz.*log2(Hinz))/L;  % find entropy
   HL=HuffLen(Hi);
   HLlen=HuffTabLen(HL);
   % find number of bits to use, store L, HL and x
   bits=6+HLlen+sum(HL.*Hi);
   if (L>=16); bits=bits+4; end;
   if (L>=272); bits=bits+4; end;
   if (L>=4368); bits=bits+5; end;
   if (L>=69904); bits=bits+5; end;
   if (L>=1118480); bits=bits+4; end;
   if Debug
      disp(['bits=',int2str(bits),'  HLlen=',int2str(HLlen),...
         '   HClen=',int2str(sum(HL.*Hi))]);
   end
else                % arguments are given, do not need to be calculated
   ent=0;
end
%
% Here we have: x, bits, L, HL, Maxx, Meanx, ent
if (L>MaxL)   % we split sequence anyway (and the easy way; in the middle)
   L1=ceil(L/2);L2=L-L1;
   x1=x(1:L1);x2=x((L1+1):L);
elseif ((Level > 0) & (L>10))
   xm=median(x); % median in MatLab is slow, could be calulated faster by using the
histogram
   x1=zeros(L,1);x2=zeros(L,1);
   x2(1)=x(1);i1=0;i2=1;
   for i=2:L
      if (x(i-1) <= xm)
         i1=i1+1; x1(i1)=x(i);
      else
         i2=i2+1; x2(i2)=x(i);
      end
   end
   x1=x1(1:i1);x2=x2(1:i2);
   % find bits1 and bits2 for x1 and x2
   L1=length(x1);L2=length(x2);
   Maxx1=max(x1);Maxx2=max(x2);
   Meanx1=mean(x1);Meanx2=mean(x2);
   Hi1=IntHist(x1,0,Maxx1);  % find the histogram
   Hi2=IntHist(x2,0,Maxx2);  % find the histogram
   HL1=hufflen(Hi1);HL2=hufflen(Hi2);
   HLlen1=HuffTabLen(HL1);
   HLlen2=HuffTabLen(HL2);
   bits1=6+HLlen1+sum(HL1.*Hi1);
   bits2=6+HLlen2+sum(HL2.*Hi2);
   if (L1>=16); bits1=bits1+4; end;
   if (L1>=272); bits1=bits1+4; end;
   if (L1>=4368); bits1=bits1+5; end;
   if (L1>=69904); bits1=bits1+5; end;
   if (L1>=1118480); bits1=bits1+4; end;
   if (L2>=16); bits2=bits2+4; end;
```

```
   if (L2>=272); bits2=bits2+4; end;
   if (L2>=4368); bits2=bits2+5; end;
   if (L2>=69904); bits2=bits2+5; end;
   if (L2>=1118480); bits2=bits2+4; end;
else
   bits1=bits;bits2=bits;
end
% Here we may have: x1, bits1, L1, HL1, Maxx1, Meanx1
% and               x2, bits2, L2, HL2, Maxx2, Meanx2
% but at least we have bits1 and bits2  (and bits)
if Debug
   disp(['Level=',int2str(Level),'  bits=',int2str(bits),'  bits1=',int2str(bits1),...
         '  bits2=',int2str(bits2),'  sum=',int2str(bits1+bits2)]);
end

if (L>MaxL)
   if Speed
      BitPos=BitPos-1;
      if (~BitPos); Byte=Byte+1; BitPos=8; end;
   else
      PutBit(1);       % indicate sequence is splitted into two
   end;
   [bits1, temp] = EncodeVector(x1);
   [bits2, temp] = EncodeVector(x2);
   bits=bits1+bits2+1;
elseif ((bits1+bits2) < bits)
   if Speed
      BitPos=BitPos-1;
      if (~BitPos); Byte=Byte+1; BitPos=8; end;
   else
      PutBit(1);       % indicate sequence is splitted into two
   end;
   [bits1, temp] = EncodeVector(x1, bits1, HL1, Maxx1, Meanx1);
   [bits2, temp] = EncodeVector(x2, bits2, HL2, Maxx2, Meanx2);
   bits=bits1+bits2+1;
else
   bits=bits+1;      % this is how many bits we are going to write
   if Debug
      disp(['EncodeVector: Level=',int2str(Level),'  ',int2str(L),...
            ' sybols stored in ',int2str(bits),' bits.']);
   end
   if Speed
      % advance Byte and BitPos without writing to y
      Byte=Byte+floor(bits/8);
      BitPos=BitPos-mod(bits,8);
      if (BitPos<=0); BitPos=BitPos+8; Byte=Byte+1; end;
   else
      % put the bits into y
      StartPos=Byte*8-BitPos;     % control variable
      PutBit(0);       % indicate that a sequence is coded
      PutVLIC(L);
      PutHuffTab(HL);
      HK=huffcode(HL);
      for i=1:L;
         n=x(i)+1;    % symbol number (value 0 is first symbol, symbol 1)
         for k=1:HL(n)
            PutBit(HK(n,k));
         end
      end
      % check if one has used as many bits as calculated
      BitsUsed=Byte*8-BitPos-StartPos;
      if (BitsUsed~=bits)
         disp(['L=',int2str(L),'  max(x)=',int2str(max(x)),'  min(x)=',int2str(min(x))]);
         disp(['BitsUsed=',int2str(BitsUsed),'  bits=',int2str(bits)]);
         error(['Huff06-EncodeVector: Logical error, (BitsUsed~=bits).']);
      end
   end
end
Level = Level + 1;
return    % end of EncodeVector

function x = DecodeVector
global y Byte BitPos
MaxL=50000;      % as in the EncodeVector function (line 216)
if GetBit
   x1=DecodeVector;
   x2=DecodeVector;
```

```
   L=length(x1)+length(x2);
   if (L>MaxL)
      x=[x1(:);x2(:)];
   else
      xm=median([x1;x2]);
      x=zeros(L,1);
      x(1)=x2(1);
      i1=0;i2=1;
      for i=2:L
         if (x(i-1) <= xm)
            i1=i1+1; x(i)=x1(i1);
         else
            i2=i2+1; x(i)=x2(i2);
         end
      end
   end
else
   L=GetVLIC;
   if (L>1)
      x=zeros(L,1);
      HL=GetHuffTab;
      if length(HL)
         Htree=HuffTree(HL);
         root=1;pos=root;
         l=0;  % number of symbols decoded so far
         while l<L
            if GetBit
               pos=Htree(pos,3);
            else
               pos=Htree(pos,2);
            end
            if Htree(pos,1)           % we have arrived at a leaf
               l=l+1;
               x(l)=Htree(pos,2)-1;   % value is one less than symbol number
               pos=root;              % start at root again
            end
         end
      else     % HL has length 0, that is empty Huffman table
         x=x+GetVLIC;
      end
   elseif L==0
      if GetBit
         % this is a Run + Value coded sequence
         x1=DecodeVector;
         x2=DecodeVector;
         % now build the actual sequence
         I=x1;      % runs
         I=I+1;
         L=length(I);  % one more than the number of values in x
         for i=2:L;I(i)=I(i-1)+I(i); end;
         x=zeros(I(L)-1,1);
         x(I(1:(L-1)))=x2;  % values
      else
         x=[];    % this was really a length 0 sequence
      end
   elseif L==1
      x=GetVLIC;
   else
      error('DecodeVector: illegal length of sequence.');
   end
end
return    % end of DecodeVector

% Functions to write and read the Huffman Table Information
% The format is defined in HuffTabLen, we repeat it here
% Function assume that the table information is stored in the following format
%        previous symbol is set to the initial value 2, Prev=2
%        Then we have for each symbol a code word to tell its length
%         '0'              - same length as previous symbol
%         '10'             - increase length by 1, and 17->1
%         '1100'           - decrease length by 1, and 0->16
%         '11010'          - increase length by 2, and 17->1, 18->2
%         '11011'          - One zero, unused symbol (twice for two zeros)
%         '111xxxx'        - set code length to CL=Prev+x (where 3 <= x <= 14)
%                            and if CL>16; CL=CL-16
%        we have 4 unused 7 bit code words, which we give the meaning
%         '1110000'+4bits - 3-18 zeros
```

```
%         '1110001'+8bits - 19-274 zeros, zeros do not change previous value
%         '1110010'+4bits - for CL=17,18,...,32, do not change previous value
%         '1111111'       - End Of Table

function PutHuffTab(HL)
global y Byte BitPos

HL=HL(:);
% if (max(HL) > 32)
%    disp(['PutHuffTab: To large value in HL, max(HL)=',int2str(max(HL))]);
% end
% if (min(HL) < 0)
%    disp(['PutHuffTab: To small value in HL, min(HL)=',int2str(min(HL))]);
% end
Prev=2;
ZeroCount=0;
L=length(HL);

for l=1:L
   if HL(l)==0
      ZeroCount=ZeroCount+1;
   else
      while (ZeroCount > 0)
         if ZeroCount<3
            for i=1:ZeroCount
               PutBit(1);PutBit(1);PutBit(0);PutBit(1);PutBit(1);
            end
            ZeroCount=0;
         elseif ZeroCount<19
            PutBit(1);PutBit(1);PutBit(1);PutBit(0);PutBit(0);PutBit(0);PutBit(0);
            for (i=4:-1:1); PutBit(bitget(ZeroCount-3,i)); end;
            ZeroCount=0;
         elseif ZeroCount<275
            PutBit(1);PutBit(1);PutBit(1);PutBit(0);PutBit(0);PutBit(0);PutBit(1);
            for (i=8:-1:1); PutBit(bitget(ZeroCount-19,i)); end;
            ZeroCount=0;
         else
            PutBit(1);PutBit(1);PutBit(1);PutBit(0);PutBit(0);PutBit(0);PutBit(1);
            for (i=8:-1:1); PutBit(1); end;
            ZeroCount=ZeroCount-274;
         end
      end
      if HL(l)>16
         PutBit(1);PutBit(1);PutBit(1);PutBit(0);PutBit(0);PutBit(1);PutBit(0);
         for (i=4:-1:1); PutBit(bitget(HL(l)-17,i)); end;
      else
         Inc=HL(l)-Prev;
         if Inc<0; Inc=Inc+16; end;
         if (Inc==0)
            PutBit(0);
         elseif (Inc==1)
            PutBit(1);PutBit(0);
         elseif (Inc==2)
            PutBit(1);PutBit(1);PutBit(0);PutBit(1);PutBit(0);
         elseif (Inc==15)
            PutBit(1);PutBit(1);Put8it(0);PutBit(0);
         else
            PutBit(1);PutBit(1);PutBit(1);
            for (i=4:-1:1); PutBit(bitget(Inc,i)); end;
         end
         Prev=HL(l);
      end
   end
end
for (i=7:-1:1); PutBit(1); end;        % the EOT codeword

return;  % end of PutHuffTab

function HL=GetHuffTab
global y 8yte BitPos

Debug=0;
Prev=2;
ZeroCount=0;
HL=zeros(10000,1);
HLi=0;
EndOfTable=0;
```

```
while ~EndOfTable
   if GetBit
      if GetBit
         if GetBit
            Inc=0;
            for (i=1:4); Inc=Inc*2+GetBit; end;
            if Inc==0
               ZeroCount=0;
               for (i=1:4); ZeroCount=ZeroCount*2+GetBit; end;
               HLi=HLi+ZeroCount+3;
            elseif Inc==1
               ZeroCount=0;
               for (i=1:8); ZeroCount=ZeroCount*2+GetBit; end;
               HLi=HLi+ZeroCount+19;
            elseif Inc==2           % HL(1) is large, >16
               HLi=HLi+1;
               HL(HLi)=0;
               for (i=1:4); HL(HLi)=HL(HLi)*2+GetBit; end;
               HL(HLi)=HL(HLi)+17;
            elseif Inc==15
               EndOfTable=1;
            else
               Prev=Prev+Inc;
               if Prev>16; Prev=Prev-16; end;
               HLi=HLi+1;HL(HLi)=Prev;
            end
         else
            if GetBit
               if GetBit
                  HLi=HLi+1;
               else
                  Prev=Prev+2;
                  if Prev>16; Prev=Prev-16; end;
                  HLi=HLi+1;HL(HLi)=Prev;
               end
            else
               Prev=Prev-1;
               if Prev<1; Prev=16; end;
               HLi=HLi+1;HL(HLi)=Prev;
            end
         end
      else
         Prev=Prev+1;
         if Prev>16; Prev=1; end;
         HLi=HLi+1;HL(HLi)=Prev;
      end
   else
      HLi=HLi+1;HL(HLi)=Prev;
   end
end
if HLi>0
   HL=HL(1:HLi);
else
   HL=[];
end

if Debug
   % check if this is a valid Huffman table
   temp=sum(2.^(-nonzeros(HL)));
   if temp ~=1
      error(['GetHuffTab: HL table is no good, temp=',num2str(temp)]);
   end
end

return;  % end of GetHuffTab

% Functions to write and read a Variable Length Integer Code word
% This is a way of coding non-negative integers that uses fewer
% bits for small integers than for large ones. The scheme is:
%   '00'   +  4 bit  - integers from 0 to 15
%   '01'   +  8 bit  - integers from 16 to 271
%   '10'   + 12 bit  - integers from 272 to 4367
%   '110'  + 16 bit  - integers from 4368 to 69903
%   '1110' + 20 bit  - integers from 69940 to 1118479
%   '1111' + 24 bit  - integers from 1118480 to 17895695
%   not supported  - integers >= 17895696 (=2^4+2^8+2^12+2^16+2^20+2^24)
```

```
function PutVLIC(N)
global y Byte BitPos
if (N<0)
   error('Huff06-PutVLIC: Number is negative.');
elseif (N<16)
   PutBit(0);PutBit(0);
   for (i=4:-1:1); PutBit(bitget(N,i)); end;
elseif (N<272)
   PutBit(0);PutBit(1);
   N=N-16;
   for (i=8:-1:1); PutBit(bitget(N,i)); end;
elseif (N<4368)
   Put8it(1);PutBit(0);
   N=N-272;
   for (i=12:-1:1); Put8it(bitget(N,i)); end;
elseif (N<69940)
   PutBit(1);PutBit(1);PutBit(0);
   N=N-4368;
   for (i=16:-1:1); Put8it(bitget(N,i)); end;
elseif (N<1118480)
   PutBit(1);PutBit(1);PutBit(1);PutBit(0);
   N=N-69940;
   for (i=20:-1:1); Put8it(bitget(N,i)); end;
elseif (N<17895696)
   Put8it(1);PutBit(1);PutBit(1);PutBit(1);
   N=N-1118480;
   for (i=24:-1:1); PutBit(bitget(N,i)); end;
else
   error('Huff06-PutVLIC: Number is too large.');
end
return

function N=GetVLIC
global y Byte BitPos
N=0;
if GetBit
   if GetBit
      if GetBit
         if GetBit
            for (i=1:24); N=N*2+GetBit; end;
            N=N+1118480;
         else
            for (i=1:20); N=N*2+GetBit; end;
            N=N+69940;
         end
      else
         for (i=1:16); N=N*2+GetBit; end;
         N=N+4368;
      end
   else
      for (i=1:12); N=N*2+GetBit; end;
      N=N+272;
   end
else
   if GetBit
      for (i=1:8); N=N*2+GetBit; end;
      N=N+16;
   else
      for (i=1:4); N=N*2+GetBit; end;
   end
end
return

% Functions to write and read a Bit
function PutBit(Bit)
global y Byte BitPos
BitPos=BitPos-1;
if (~BitPos); Byte=Byte+1; BitPos=8; end;
y(Byte) = bitset(y(Byte),BitPos,Bit);
return

function Bit=GetBit
global y Byte BitPos
BitPos=BitPos-1;
if (~BitPos); Byte=Byte+1; BitPos=8; end;
Bit=bitget(y(Byte),BitPos);
return;
```

```
% this function is a variant of the standard hist function
function Hi=IntHist(W,i1,i2);
W=W(:);
L=length(W);
Hi=zeros(i2-i1+1,1);
if (i2-i1)>50
   for l=1:L
      i=W(l)-i1+1;
      Hi(i)=Hi(i)+1;
   end
else
   for i=i1:i2
      I=find(W==i);
      Hi(i-i1+1)=length(I);
   end
end
return;


function HK = HuffCode(HL,Display)
% HuffCode    Based on the codeword lengths this function find the Huffman codewords
%
% HK = HuffCode(HL,Display);
% HK = HuffCode(HL);
% -------------------------------------------------------------------
% Arguments:
%  HL     length (bits) for the codeword for each symbol
%         This is usually found by the hufflen function
%  HK     The Huffman codewords, a matrix of ones or zeros
%         the code for each symbol is a row in the matrix
%         Code for symbol S(i) is: HK(i,1:HL(i))
%         ex: HK(i,1:L)=[0,1,1,0,1,0,0,0] and HL(i)=6 ==>
%             Codeword for symbol S(i) = '011010'
%  Display==1  ==> Codewords are displayed on screen, Default=0
% -------------------------------------------------------------------

%-------------------------------------------------------------------
% Copyright (c) 1999.  Karl Skretting.  All rights reserved.
% Hogskolen in Stavanger (Stavanger University), Signal Processing Group
% Mail:  karl.skretting@tn.his.no   Homepage:  http://www.ux.his.no/~karlsk/
%
% HISTORY:
% Ver. 1.0  25.08.98  KS: Function made as part of Signal Compression Project 98
% Ver. 1.1  25.12.98  English version of program
%-------------------------------------------------------------------

if nargin<1
   error('huffcode: see help.')
end
if nargin<2
  Display = 0;
end
if (Display ~= 1)
   Display = 0;
end

N=length(HL);
L=max(HL);
HK=zeros(N,L);
[HLs,HLi] = sort(HL);
Code=zeros(1,L);
for n=1:N
   if (HLs(n)>0)
      HK(HLi(n),:) = Code;
      k = HLs(n);
      while (k>0)                    % actually always!  break ends loop
         Code(k) = Code(k) + 1;
         if (Code(k)==2)
            Code(k) = 0;
            k=k-1;
         else
            break
         end
      end
   end
end
```

```
if Display
   for n=1:N
      Linje = ['  Symbol ',int2str(n)];
      for i=length(Linje):15
         Linje = [Linje,' '];
      end
      Linje = [Linje,'  gets code: '];
      for i=1:HL(n)
         if (HK(n,i)==0)
            Linje = [Linje,'0'];
         else
            Linje = [Linje,'1'];
         end
      end
      disp(Linje);
   end
end

return;


function HL = HuffLen(S)
% HuffLen     Find the lengths of the Huffman code words
% Based on probability (or number of occurences) of each symbol
% the length for the Huffman codewords are calculated.
%
% HL = hufflen(S);
% ------------------------------------------------------------------
% Arguments:
%  S  a vector with number of occurences or probability of each symbol
%     Only positive elements of S are used, zero (or negative)
%     elements get length 0.
%  HL length (bits) for the codeword for each symbol
% ------------------------------------------------------------------
% Example:
% hufflen([1,0,4,2,0,1])  =>  ans = [3,0,1,2,0,3]
% hufflen([10,40,20,10])  =>  ans = [3,1,2,3]

%------------------------------------------------------------------------
% Copyright (c) 1999.  Karl Skretting.  All rights reserved.
% Hogskolen in Stavanger (Stavanger University), Signal Processing Group
% Mail:  karl.skretting@tn.his.no   Homepage:  http://www.ux.his.no/~karlsk/
%
% HISTORY:
% Ver. 1.0  28.08.98  KS: Function made as part of Signal Compression Project 98
% Ver. 1.1  25.12.98  English version of program
% Ver. 1.2  28.07.99  Problem when length(S)==1 was corrected
% Ver. 1.3  22.06.00  KS: Some more exceptions handled
%------------------------------------------------------------------------

if nargin<1
   error('HuffLen: see help.')
end
% some checks and exceptions
if (length(S)==0)          % ver 1.2
   warning('HuffLen: Symbol sequence is empty.');   % a warning is appropriate
   HL=0;
   return;
end
I=find(S<0);
S(I)=0;
if (sum(S)==max(S))
   disp('HuffLen: Only one symbol.');   % a message is appropriate
   HL=zeros(size(S));       % no Huffman code is needed
   return;
end

%  Algorithm "explained" in Norwegian:
%   En bygger opp "treet" ved å legge sammen de to nodene som har
%   minst C, C teller hvor mange verdier som er samlet under denne noden
%   De N første i C er bladene, de andre er noder med to andre noder (blad)
%   under seg, men en trenger ikke nøyaktig hvordan treet er under hver node
%   Det en trenger er for hvert blad å vite hvilken node som er øverst
%   i treet den er festet på, dette er lagret i Top, startverdier er her
%   bladet selv (blad ennå ikke samlet i tre)
```

```
%   Si er indekser for toppnodene i C, de er sortert etter hvor mange
%   verdier (count) for hver node. Kun Si(1:last) er interessante,
%   siden en kun har "last" trær. (for hver gang hovedløkka kjører
%   samles to trær til et tre, alle blad som hører til hvert av disse
%   trærne før kodeordeslengden, HL, øket med en, og en må oppdatere hvilken
%   node som nå er toppen for dette bladet, Top(I) settes.

HL=zeros(size(S));
S=S(:);
Ip=find(S>0);       % index of positive elements
Sp=S(Ip);           % the positive elements of S

N=length(Sp);       % elements in Sp vector
HLp=zeros(size(Sp));
C=[Sp(:);zeros(N-1,1)];  % count or weights for each "tree"
Top=1:N;                 % the "tree" every symbol belongs to
[So,Si]=sort(-Sp);       % Si is indexes for descending symbols
last=N;                  % Number of "trees" now
next=N+1;                % next free element in C
while (last > 1)
   % the two smallest "trees" are put together
   C(next)=C(Si(last))+C(Si(last-1));
   I=find(Top==Si(last));
   HLp(I)=HLp(I)+1;   % one extra bit added to elements in "tree"
   Top(I)=next;
   I=find(Top==Si(last-1));
   HLp(I)=HLp(I)+1;   % and one extra bit added to elements in "tree"
   Top(I)=next;
   last=last-1;
   Si(last)=next;
   next=next+1;
   % Si shall still be indexes for descending symbols or nodes
   count=last-1;
   while ((count> 0) & (C(Si(count+1)) >= C(Si(count))))
      temp=Si(count);
      Si(count)=Si(count+1);
      Si(count+1)=temp;
      count=count-1;
   end
end

HL(Ip)=HLp;
return;



function HLlen = HuffTabLen(HL)
% HuffTabLen  Find how many bits we need to store the Huffman Table information
%
% HLlen = HuffTabLen(HL);
%-------------------------------------------------------------------------
% arguments:
%  HL       The codeword lengths, as returned from HuffLen function
%           This should be a vector of integers
%           where  0 <= HL(i) <= 32, 0 is for unused symbols
%           We then have max codeword length is 32
%  HLlen    Number of bits needed to store the table
%-------------------------------------------------------------------------

% Function assume that the table information is stored in the following format
%       previous code word length is set to the initial value 2
%       Then we have for each symbol a code word to tell its length
%         '0'             - same length as previous symbol
%         '10'            - increase length by 1, and 17->1
%         '1100'          - reduce length by 1, and 0->16
%         '11010'         - increase length by 2, and 17->1, 18->2
%         '11011'         - One zero, unused symbol (twice for two zeros)
%         '111xxxx'       - set code length to CL=Prev+x (where 3 <= x <= 14)
%                           and if CL>16; CL=CL-16
%       we have 4 unused 7 bit code words, which we give the meaning
%         '1110000'+4bits - 3-18 zeros
%         '1110001'+8bits - 19-274 zeros, zeros do not change previous value
%         '1110010'+4bits - for CL=17,18,...,32, do not change previous value
%         '1111111'       - End Of Table

%-------------------------------------------------------------------------
% Copyright (c) 1999.  Karl Skretting.  All rights reserved.
```

```
% Hogskolen in Stavanger (Stavanger University), Signal Processing Group
% Mail:  karl.skretting@tn.his.no   Homepage:  http://www.ux.his.no/~karlsk/
%
% HISTORY:
% Ver. 1.0  18.08.99  KS, function made as an own m-file (another version
%                     is included in the Huff04 m-file)
%           25.08.99  KS: now we use this format also in Huff04
%                     if you change here, remember to also update Huff04!!
% Ver. 1.2  26.08.99  KS: Reduced number of bits used for zeros (increased for +2)
% Ver. 1.3  20.06.00  KS: Removed the KeepStatistics lines
% Ver. 1.4  18.01.01  KS: Removed error message if HL is out of (normal) range.
% Ver. 1.5  21.08.01  KS: Allow HL to be in range  0 <= HL(i) <= 32
%-------------------------------------------------------------------------

Mfile='HuffTabLen';
% KeepStatistics=0;  % we may want to keep statistics to see wether the chosen
                     % code words are well suited
if (nargin ~= 1);
   error([Mfile,': function must have one input arguments, see help.']);
end
if (nargout ~= 1);
   error([Mfile,': function must have one output arguments, see help.']);
end
HL=HL(:);

if (max(HL) > 32)
   disp([Mfile,': To large value in HL, max(HL)=',int2str(max(HL))]);
end
if (min(HL) < 0)
   disp([Mfile,': To small value in HL, min(HL)=',int2str(min(HL))]);
end

Prev=2;
HLlen=0;
ZeroCount=0;
% if KeepStatistics; load HuffTabLenStat; end;   %
IncStat=zeros(16,1);RunStat=zeros(512,1);

L=length(HL);
for l=1:L
   if HL(l)==0
      ZeroCount=ZeroCount+1;
   else
      % if (ZeroCount & KeepStatistics)
      %    j=min([512,ZeroCount]); RunStat(j)=RunStat(j)+1;
      % end
      while (ZeroCount > 0)
         if ZeroCount<3; HLlen=HLlen+5*ZeroCount; ZeroCount=0;
         elseif ZeroCount<19; HLlen=HLlen+11; ZeroCount=0;
         elseif ZeroCount<275; HLlen=HLlen+15; ZeroCount=0;
         else HLlen=HLlen+15; ZeroCount=ZeroCount-274; end;
      end
      if HL(l)>16
         HLlen=HLlen+11;
      else
         Inc=HL(l)-Prev;
         if Inc<0; Inc=Inc+16; end;
         % if KeepStatistics; j=Inc+1; IncStat(j)=IncStat(j)+1; end;
         if (Inc==0); HLlen=HLlen+1;
         elseif (Inc==1); HLlen=HLlen+2;
         elseif (Inc==2); HLlen=HLlen+5;
         elseif (Inc==15); HLlen=HLlen+4;
         else HLlen=HLlen+7;
         end
         Prev=HL(l);
      end
   end
end
HLlen=HLlen+7;       % the EOT codeword

% if KeepStatistics; save HuffTabLenStat IncStat RunStat; end;

return;  % end of HuffTabLen


function Htree = HuffTree(HL,HK)
% HuffTree    Make the Huffman-tree from the lengths of the Huffman codes
```

```
% The Huffman codes are also needed, and if they are known
% they can be given as an extra input argument
%
% Htree = HuffTree(HL,HK);
% Htree = HuffTree(HL);
% ------------------------------------------------------------------
% Arguments:
%  HL     length (bits) for the codeword for each symbol
%         This is usually found by the hufflen function
%  HK     The Huffman codewords, a matrix of ones or zeros
%         the code for each symbol is a row in the matrix
%  Htree  A matrix, (N*2)x3, representing the Huffman tree,
%         needed for decoding. Start of tree, root, is Htree(1,:).
%         Htree(i,1)==1 indicate leaf and Htree(i,1)==0 indicate branch
%         Htree(i,2) points to node for left tree if branching point and
%         symbol number if leaf. Note value is one less than symbol number.
%         Htree(i,3) points to node for right tree if branching point
%         Left tree is '0' and right tree is '1'
% ------------------------------------------------------------------

%-------------------------------------------------------------------
% Copyright (c) 1999.  Karl Skretting.  All rights reserved.
% Hogskolen in Stavanger (Stavanger University), Signal Processing Group
% Mail:  karl.skretting@tn.his.no   Homepage:  http://www.ux.his.no/~karlsk/
%
% HISTORY:
% Ver. 1.0  25.08.98  KS: Function made as part of Signal Compression Project 98
% Ver. 1.1  25.12.98  English version of program
%-------------------------------------------------------------------

if nargin<1
   error('hufftree: see help.');
end
if nargin<2
  HK = huffcode(HL);
end
N=length(HL);       % number of symbols

Htree=zeros(N*2,3);
root=1;
next=2;
for n=1:N
   if HL(n)>0
      % place this symbol correct in Htree
      pos=root;
      for k=1:HL(n)
         if ((Htree(pos,1)==0) & (Htree(pos,2)==0))
            % it's a branching point but yet not activated
            Htree(pos,2)=next;
            Htree(pos,3)=next+1;
            next=next+2;
         end
         if HK(n,k)
            pos=Htree(pos,3);     % goto right branch
         else
            pos=Htree(pos,2);      % goto left branch
         end
      end
      Htree(pos,1)=1;   % now the position is a leaf
      Htree(pos,2)=n;   % and this is the symbol number it represent
   end
end
if N==1
   Htree(1,3)=2;
end

return
```

```
function [out,scale] = quantiz(in,nbit)

% nbit - The number of bits per value

%Obtain the maximum values in the coefficient matrix
maxc = max(max(abs(in)));

% Quantization levels
% Lowest level
lowlvl = -1*(2^(nbit-1));
% Highest level
highlvl = 2^(nbit-1) - 1;

% Scale
scale = highlvl/maxc;

% Quantize the input coefficient matrix to nbit bits per coefficient
out = in * scale;
out = floor(out);



function out = dequantiz(in,scale)

% Quantize the input coefficient matrix to nbit bits per coefficient
out = in / scale;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author : Vooi Voon YAP, Middlesex University, London                                       %
%--------------------------------------------------------------------------------------------%
% First written: 26/11/2004                                                                  %
% Last Update:   27/11/2004                                                                  %
%--------------------------------------------------------------------------------------------%
% Main Function : MPICompressorV1.m                                                          %
%--------------------------------------------------------------------------------------------%
% Purpose :                                                                                  %
% This is a wavelet-based image compression program written for my PhD research.            %
% Compressed colour images using crop-based techniques for mobile telephones.               %
% Blockprocessing used to process backgroung image. Pixels in each block is averaged.       %
% Image is resize at source.                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Get the image %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Orig_Img = imread('D:\matlab6p5\work\images\natural\bird.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\marcie.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\chiliq01.bmp');

% Orig_Img=imread('D:\matlab6p5\work\images\natural\mandrill.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\chiliq02.bmp');
%  Orig_Img=imread('D:\matlab6p5\work\images\natural\flower02_256.bmp');

 Orig_Img = imread('D:\matlab6p5\work\images\synthetic\clipart10_256.bmp');
% -------------------------------- Select dimension -------------------------------- %
Model = 'Nokia 3220';
% Model = 'Motorola E398';
% Model = 'Sony Ericsson T40';
% Model = 'Sony Ericsson T68';
% Model = 'MAX Q';

[w h] = SelectDimension(Model);
% -------------------------------- Resize image ------------------------------------ %

% w = 128;
% h = 128;
ScrnSize=ScreenSize(w,h);
Orig_Img = imresize(Orig_Img,[ScrnSize ScrnSize],'nearest');
figure(1), imshow(Orig_Img),title('Original image');
Orig_Img = double(Orig_Img);
% -------------------------------------------------------------------------------- %

Y  = 0.299*Orig_Img(:,:,1) + 0.587*Orig_Img(:,:,2) + 0.114*Orig_Img(:,:,3);
```

```
Cb = -0.16875*Orig_Img(:,:,1) + -0.33126*Orig_Img(:,:,2) + 0.5*Orig_Img(:,:,3);
Cr = 0.5*Orig_Img(:,:,1) + -0.41869*Orig_Img(:,:,2) + -0.08131*Orig_Img(:,:,3);

% ---------------------------------- Crop image ---------------------------------- %
% [CropYimg, rect] = imcrop(Y,[40 25 60 75]); % for bird
% [CropYimg, rect] = imcrop(Y,[15 20 90 100]); % for chiliq01, marcie

[CropYimg, rect] = imcrop(Y,[20 5 90 105]); % for mandrill
% [CropYimg, rect] = imcrop(Y,[25 5 90 105]); % for chiliq02
% [CropYimg, rect] = imcrop(Y,[45 40 40 60]); % for flower256

figure(2), imshow(uint8(CropYimg)),title('Cropped image');
% ------------------------------------------------------------------------------- %

% ------------------------------ Find average of mask ------------------------------ %
MaskAveValue = FindMaskAve(CropYimg,Y,rect);
% ------------------------------------------------------------------------------- %

% ---------------------------------- Create mask ---------------------------------- %
Mask = CropYimg;
Mask = (double(Mask)).*0;
Mask = (~Mask).*MaskAveValue; % 105 original value for marcie ...

% resize image %
xMaskSize = size(Mask,1);
yMaskSize = size(Mask,2);
cropMask = imcrop(Mask,[0 0 yMaskSize-7 xMaskSize-7]); % the offset is different
% ------------------------------------------------------------------------------- %

% ------------------------------ background image ------------------------------ %
% Now replace pixels in the background image with the object image
newI = Y;

% Define where to place the object image in the background image
colshift = rect(1); % the offset is set at 3 for this version
rowshift = rect(2); % the offset is set at 3 for this version

% Perform the actual indexing to replace the background image pixels with the object
newI((1:size(cropMask,1))+rowshift, (1:size(cropMask,2))+colshift, :) = cropMask;
figure(3),imshow(uint8(newI));title('New Image');
% ---------------------------- Divide image into blocks ---------------------------- %
[l, m, h] = DefineFreqZones(Orig_Img);
blksz = SelectBlockSz (l,m,h);
IblkSz = blksz;

fun = inline('uint8(round(mean2(x)))');

AnewI = blkproc(newI,[IblkSz IblkSz],fun);
figure(4),imshow(AnewI);title('Averaged Image');
% ------------------------------------------------------------------------------- %

% ---------------------------- Encode background image ---------------------------- %
% [qAnewI,scale] = quantiz(double(AnewI),8.1);
[i,j] = size(AnewI);
enYOutput = EntropyCoder(double(AnewI),'encode',i,j);
% ------------------------------------------------------------------------------- %
[wName1 wName2] = SelectWavelet(Orig_Img)
% wName1 = 'bior4.4';
% wName2 = 'haar';
YLevel = 3;
CbCrLevel = 4;

% ------------------------------ compress images ------------------------------ %
[YRecon, YEncoded, yTotalCoeff, yNonZeroCoeff]...
    = YComp(CropYimg, wName1, YLevel);

[CbRecon, CbcomEncoded, CbcomTotalCoeff, CbcomNonZeroCoeff]...
    = CbComp(Cb, wName2, CbCrLevel);

[CrRecon, CrcomEncoded, CrcomTotalCoeff, CrcomNonZeroCoeff]...
    = CrComp(Cr, wName2, CbCrLevel);

% figure(5),imshow(uint8(CompImage));title('Reconstructed Image');
cImage = YRecon;

% ------------------------------------------------------------------------------- %
% ------------------------------------------------------------------------------- %
```

```
% ---------------------------- Transmit encoded image ---------------------------- %
% The block size and the size of orginal image should be transmitted together with    %
% encoded image.                                                                      %
eOutput = size(enYOutput,1);
% fprintf (1, '\nEncoded output:      %d\n', eOutput);
imgSz = ScrnSize;
% ------------------------------------------------------------------------------- %

% ------------------------------ Decode encoded image ------------------------------ %
decImage = EntropyCoder(enYOutput,'decode',i,j);
% ------------------------------------------------------------------------------- %

% ------------------------------ Reconstruct image -------------------------------- %

rowLength = size(decImage,2);              % szA is the no. of elements in a row
decImage = imagetovector(decImage);        % convert image to row vector

tic;
list = VectToImg(decImage,rowLength,IblkSz);
toc;

recImage = vectortoimage(list,imgSz,imgSz); % convert row vector to image

% Now replace pixels in the background image with the object image
nYImage = uint8(recImage);

% Define where to place the object image in the scene
colshift = rect(1);
rowshift = rect(2);

nYImage((1:size(YRecon,1))+rowshift, (1:size(YRecon,2))+colshift, :) = YRecon;

Y = nYImage;
Cb = CbRecon;
Cr = CrRecon;

Y = double(Y);
Cb = double(Cb);
Cr = double(Cr);

R = Y + 1.402.*Cr;
G = Y + (-0.34413.*Cb) + (-0.71414.*Cr);
B = Y + 1.772.*Cb ;

Final_Img(:,:,1) = R;
Final_Img(:,:,2) = G;
Final_Img(:,:,3) = B;
figure(5),imshow(uint8(Final_Img));title('Reconstructed Image');

fprintf (1, 'Wavelet Type 1:           %s\n', wName1);
fprintf (1, 'Wavelet Type 2:           %s\n', wName2);
fprintf (1, '\nScreen size:                    %dx%d\n', ScrnSize, ScrnSize);
fprintf (1, '\nBlock size:                     %d\n', IblkSz);
fprintf (1, '\nEncoded output:                %d\n', eOutput);
fprintf (1, '\nY cropped image encoded output:   %2.2f\n', size(YEncoded,1));
fprintf (1, '\nCb encoded output:               %2.2f\n', size(CbcomEncoded,1));
fprintf (1, '\nCr encoded output:               %2.2f\n', size(CrcomEncoded,1));

Final_Img = uint8(Final_Img); % needs to be converted back to uint8 before writing to a
image file
imwrite(Final_Img,'D:\Matlab6p5\work\Images\cDummyImg.bmp');


function [width, height] = SelectDimension(model)

switch model
    case 'Nokia 3220'
       fprintf (1, 'Model:      %s\n', model);
       w = 128;
       h = 128;
    case 'Motorola E398'
       fprintf (1, 'Model:      %s\n', model);
       w = 176;
       h = 220;
    case 'Sony Ericsson T40'
       fprintf (1, 'Model:      %s\n', model);
```

```
        w = 128;
        h = 160;
    case 'Sony Ericsson T68'
        fprintf (1, 'Model:      %s\n', model);
        w = 101;
        h = 80;
    otherwise
        fprintf (1, 'Model unknown:      %s\n', model);
 end
width = w;
height = h;


function newList = VectToImg(in,rLength,IblkSz)

szA = rLength;                    % szA is the no. of elements in a row
h=in;

% blkSz = IblkSz*IblkSz;
blkSz = IblkSz;                       % block size

e = 1;                                % position of element
Initial_e = e;                        % temporary storage
eTrack = 1;                           % track the elements in a block
stMark = 1;                           % start of element to be repeated
enMark = blkSz;                       % end of element to be repeated
eBuffer = 0;

m = 1;                                % initial condition
row = 1;                              % initial condition
Height = 1;                           % Height of input matrix. initial condition

while Height <= szA                   % repeat szA times
%
       while row <= blkSz                % repeat blkSz times

        while eTrack <= szA           % repeat szA times.

            for m = stMark:enMark     % get element and repeat blkSz times
                list(m) = h(e);
            end
            stMark = stMark + blkSz;
            enMark = enMark + blkSz;
            e = e+1;
            eBuffer = e;
            eTrack = eTrack + 1;

         end
         row = row +1;
         e = Initial_e;
         eTrack = 1;

       end
    e = eBuffer;
    Initial_e = eBuffer;
    row = 1;
    Height = Height + 1;
end
% reconImage = vectortoimage(list,ImgSz,ImgSz);
newList = list;


function blksz = SelectBlockSz (l,m,h)

L = l;
M = m;
H = h;
Flag = 1;

if L > M
   if (M+H) >= 60
      Select ='M'
      Flag = 0;
   else
      Select = 'L'
      Flag = 0;
   end
```

```
end

if Flag ~= 0
   if M > H
      if H  >= 35
         Select = 'H'
         Flag = 0;
      end
      if Flag ~= 0
         Select = 'M'
      end
   else
      Select = 'H'
   end
end


if Select == 'L'
   blksize = 16;
end
if Select == 'M'
   blksize = 8;
end
if Select == 'H'
   blksize = 4;
end

blksz = blksize;


function [wType1, wType2, iType] = SelectWavelet(i)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function name: SelectWavelet.m                                                     %
% Purpose: Select a wavelet for an image. Needs ComputeEntropy.m                     %
% Date: 12/10/2003                                                                   %
% ComputeEntropy.m written by Nikola Sprljan                                         %
%                     University of Zagreb, Faculty of Elec. Eng. & Computing, Croatia %
% Reference: Image Analyzer - Educational Tool                                       %
%            Grgic, M, Sprljan, N, Zovko-Cinlar, B                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clear
% clc
%
% figure(1)
% imshow(A);

A=double(i);

[E00,RE0,E,RH0,RH1,RH2]=ComputeEntropy(A(:,:,1));
fprintf ('\nRed H0:      %3.8f', RH0);
fprintf ('\nRed H1:      %3.8f', RH1);
fprintf ('\nRed H2:      %3.8f\n', RH2);

[E00,GE0,E,GH0,GH1,GH2]=ComputeEntropy(A(:,:,2));
fprintf ('\nGreen H0:    %3.8f', GH0);
fprintf ('\nGreen H1:    %3.8f', GH1);
fprintf ('\nGreen H2:    %3.8f\n', GH2);

[E00,BE0,E,BH0,BH1,BH2]=ComputeEntropy(A(:,:,3));
fprintf ('\nBlue H0:     %3.8f', BH0);
fprintf ('\nBlue H1:     %3.8f', BH1);
fprintf ('\nBlue H2:     %3.8f\n', BH2);

if ((RH1 <= 5)&(GH1 <= 5)&(BH1 <= 5))
   wName1 = 'haar';
   wName2 = 'haar';
   imageType = 'clipart'
else
   wName1 = 'bior4.4';
   wName2 = 'haar';
end;

wType1=wName1;
wType2=wName2;
iType = imageType;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Vooi Voon YAP, Middlesex University, London                              %
% First written: 22/09/2004                                                        %
% Last Update:                                                                     %
%----------------------------------------------------------------------------------%
% Main Function : DefineFreqZones.m                                                %
%----------------------------------------------------------------------------------%
% Purpose :                                                                        %
% Analyse an image using frequency domain and block processing method.             %
% The image is first divided into 16x16 blocks and the 2D-FFT is applied           %
% to each block to obtain the log power spectra of each block.                     %
% This version uses three zones to classify the image.                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [l, m, h] = DefineFreqZones(in)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Get the image %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Orig_Img = imread('D:\matlab6p5\work\images\natural\bird.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\marcie.bmp');
% Orig_Img = imread('D:\matlab6p5\work\images\natural\chiliq01.bmp');
% Orig_Img=imread('D:\matlab6p5\work\images\natural\barboon.bmp'); % testing only

% figure(1), imshow(Orig_Img),title('Original image');

Orig_Img = double(in);
% ---------------------------------------------------------------------------------- %
[width height layer] = size(Orig_Img);
if layer == 3
   Y  = 0.299*Orig_Img(:,:,1) + 0.587*Orig_Img(:,:,2) + 0.114*Orig_Img(:,:,3);
   Cb = -0.16875*Orig_Img(:,:,1) + -0.33126*Orig_Img(:,:,2) + 0.5*Orig_Img(:,:,3);
   Cr = 0.5*Orig_Img(:,:,1) + -0.41869*Orig_Img(:,:,2) + -0.08131*Orig_Img(:,:,3);
else
   Y = Orig_Img;
end

% [CropYimg, rect] = imcrop(Y,[80 45 120 155]); % for bird
[CropYimg, rect] = imcrop(Y,[30 40 180 190]); % for gmarcie, gchiliq01

% figure(2), imshow(uint8(CropYimg)),title('Cropped image');
% ---------------------------------------------------------------------------------- %

% -------------------------------- Create mask ------------------------------------ %
Mask = CropYimg;
Mask = (double(Mask)).*0;

% resize image %
xMaskSize = size(Mask,1);
yMaskSize = size(Mask,2);
cropMask = imcrop(Mask,[0 0 yMaskSize-7 xMaskSize-7]);
% ---------------------------------------------------------------------------------- %

% ------------------------------ background image -------------------------------- %
% Now replace pixels in the background image with the object image
newI = Y;

% Define where to place the object image in the background image
colshift = (rect(1)+3);
rowshift = (rect(2)+3);

% Perform the actual indexing to replace the background image pixels with the object
newI((1:size(cropMask,1))+rowshift, (1:size(cropMask,2))+colshift, :) = cropMask;
% figure(3),imshow(uint8(newI));title('New Image');

A = newI;

Y=double(A);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fun1 = @fft2;
fun2 = @fftshift;

no = 16;                          % define block szie

% ---------------------------- Perform block FFT ---------------------------- %
Y1 = blkproc(Y,[no no],fun1);
YFFTAveMag = blkproc(Y1,[no no],'sum(sum(abs(Y1)))./(16*16)');
```

```
YFFTAveMag1D = imagetovector(YFFTAveMag);
YFFTAveMag1Dr = round(YFFTAveMag1D);

Y2 = blkproc(Y1,[no no],fun2);
LogY=log(abs(Y2));
% ------------------------------------------------------------------------ %

% ------------------------------- Remove zeros ---------------------------- %
YFFTAveMag1Dr = RidZeros(YFFTAveMag1Dr); % get rid of the zeros
index=length(YFFTAveMag1Dr);
% ------------------------------------------------------------------------ %

Numbers=1:index;
% Numbers=1:256;
[YfreqList]=SortValues(YFFTAveMag1Dr);

% ------------------------------ Sort out the zones ------------------------ %
YValues = 0:max(YFFTAveMag1Dr);
YNumbers = size(YValues,2);
ElementsPerZone = round(YNumbers/3);

Y1Zone = sum(YfreqList(1:ElementsPerZone));
Zone2Begin = ElementsPerZone+1;
Zone2End =  Zone2Begin+ElementsPerZone;
Y2Zone = sum(YfreqList(Zone2Begin:Zone2End));
Zone3Begin = Zone2End+1;
Y3Zone = sum(YfreqList(Zone3Begin:YNumbers));

Yk = kurtosis(YFFTAveMag1Dr);
Ys = skewness(YFFTAveMag1Dr);

L = (Y1Zone/(Y1Zone+Y2Zone+Y3Zone))*100;
M = (Y2Zone/(Y1Zone+Y2Zone+Y3Zone))*100;
H = (Y3Zone/(Y1Zone+Y2Zone+Y3Zone))*100;

l = L;
m = M;
h = H;
```

Appendix 5

This appendix contains the questionnaire use in the subjective image evaluation.

Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality.
2	Fine	An image of high quality. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very poor image, but still acceptable. Objectionable artefacts are highly visible.
6	Unusable	An image so bad that it is unacceptable.

Give a value to the image displayed on the mobile phone screen using the rating scale given in the above table. Record your value on a separate score sheet.

DO NOT write on this sheet.

Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality.
2	Fine	An image of high quality. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very poor image, but still acceptable. Objectionable artefacts are highly visible.
6	Unusable	An image so bad that it is unacceptable.

Give a value to the image displayed on the mobile phone screen using the rating scale given in the above table. Record your value on a separate score sheet.

DO NOT write on this sheet.

Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality.
2	Fine	An image of high quality. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very poor image, but still acceptable. Objectionable artefacts are highly visible.
6	Unusable	An image so bad that it is unacceptable.

Give a value to the image displayed on the mobile phone screen using the rating scale given in the above table. Record your value on a separate score sheet.

DO NOT write on this sheet.

Value	*Rating*	*Description*
1	Excellent	An image of extremely high quality.
2	Fine	An image of high quality. Artefacts are not objectionable.
3	Passable	An image of acceptable quality. Artefacts are not objectionable.
4	Marginal	An image of poor quality. Artefacts are somewhat objectionable.
5	Inferior	A very poor image, but still acceptable. Objectionable artefacts are highly visible.
6	Unusable	An image so bad that it is unacceptable.

Give a value to the image displayed on the mobile phone screen using the rating scale given in the above table. Record your value on a separate score sheet.

DO NOT write on this sheet.